Creating an Interactive Algorithm for Improved Management of Difficult Airways in Micrognathic and Retrognathic Infants

by
Lauren Rakes

A thesis submitted to Johns Hopkins University
in conformity with the requirements for the degree of
Master of Arts in Medical and Biological Illustration.

Baltimore, Maryland
March, 2018

© 2018 Lauren Rakes
All Rights Reserved

Abstract

Newborns with conditions such as Pierre Robin sequence, Treacher Collins syndrome, and Goldenhar syndrome are born with underdeveloped mandibles, a condition known as “micrognathia”. Micrognathic and retrognathic (posterior positioned jaw) patients face challenges and risk of injury when anesthesia is required, due to a hypoplastic mandible, posterior displacement of the tongue, and other facial anomalies that ultimately cause airway obstruction and difficult intubation.

Currently, a “Difficult Airway Algorithm” is used to guide physicians to make the best choices during adult intubation, depending on the particular challenges and dynamics of the case. However, there are no widely available or accepted difficult airway algorithms for micrognathic infants. Furthermore, simulation models exist for practicing intubation on newborns, but only one model is available with micrognathia and is not readily available at many medical teaching facilities. Lack of training resources and contact with patients with this condition prevent physicians from acquiring the experience necessary to adequately care for these infants.

An interactive web application was created to provide an accessible way to increase exposure to the anatomy of micrognathic newborns and the recommended techniques for intubation. A unique interactive algorithm within the application was developed specifically for the treatment and airway management of children with micrognathia. Interactivity allows the user to explore different paths within the algorithm and learn additional information through text, illustration, and animation. Additionally, a separate section of the application allows for exploration and deconstruction of a 3D model of a micrognathic patient, allowing for reinforcement of anatomical and spatial concepts. Together the unique interactive flowchart, the 3D anatomy viewer, and an additional background information section with supplemental text and illustrations, provide a useful overview for users that anticipate the need to manage a micrognathic airway. This project will contribute to the safety of these children inside and outside the operating room by increasing clinician’s exposure to the different options available during intubation. The workflow and foundation of this project has the potential to be expanded and applied to a variety of medical scenarios.

Lauren Rakes

Chairpersons of the Supervisory Committee

Deborah Schwengel, MD, MEd, Preceptor

Assistant Professor of Anesthesiology, Critical Care Medicine and Pediatrics

The Johns Hopkins University School of Medicine

ACCM Director of Education Research

Division of Pediatric Anesthesiology

The Johns Hopkins Hospital

Jonathan Walsh, MD, Preceptor

Assistant Professor of Otolaryngology, Division of Pediatric Otolaryngology

The Johns Hopkins University School of Medicine

Otolaryngology Head and Neck Surgery

The Johns Hopkins Hospital

David Rini, MFA, CMI, FAMI, Department Advisor

Associate Professor, Department of Art as Applied to Medicine

The Johns Hopkins School of Medicine

Acknowledgement

This project would not have been possible without the support of many amazing people. It's been wonderful to have this opportunity to work with and be surrounded by these bright minds. I would like to extend my sincerest thanks to:

David Rini, my advisor and Associate Professor in the Art as Applied to Medicine Department for his innovative ideas, helpful feedback, and guidance throughout this project and every other endeavour.

Dr. Deborah Schwengel from the Department of Pediatric Anesthesiology and Critical Care Medicine, and **Dr. Jonathan Walsh** from the Department of Otolaryngology and Head and Neck Surgery, for acting as my preceptors for this thesis. Your encouragement, expertise, and direction were crucial in the success of this project.

Corrine Sandone, Gary Lees, Jennifer Fairman, Tim Phelps, Juan Garcia, Ian Suk, Lydia Gregg, Norman Barker, Anne Altemus, Donald Bliss, Mike Linkinhoker, Sandra Gabelli, and Virginia Ferrante, for teaching me so many valuable lessons and skills during my time at Johns Hopkins.

Dacia Balch, for taking care of my classmates and I from the very beginning, and all the things you do to make the Art as Applied to Medicine Department feel like home.

Carol Pfeffer, for her frequent check-ins and optimism.

Sarah Poynton, for her immense help in grant applications.

My parents: **Bev Rakes** and **Gary Rakes,** my siblings: **Kelsey Sorge** and **Jack Rakes**, my brother-in-law: **Dylan Sorge** and my nephew: **Oscar Sorge**. Thank you for so many years of love and creativity. Your encouragement is the reason I chose to become a medical illustrator. Oscar, thank you for being a reference for many illustrations in this project.

Kevin Hensley, for your love and unwaivering support through every decision in my life.

My classmates: **Tziporah Thompson, Shawna Snyder, Hillary Wilson, Amanda Slade,** and **Mary Shi** for their constant companionship, friendship, and inspiration over these past two years.

Table of Contents

List of Tables

Figure Index

List of Abbreviations

PRS: Pierre Robin Sequence

LMA: Laryngeal Mask Airway

CT: Computed Tomography

WebGL: Web Graphics Library

DICOM: Digital Imaging and Communications in Medicine

PDF: Portable Document Format

TIFF: Tagged Image Format File

Bool OR **boole:** Boolean operation

"Bool" is used in Unity, "boole" is used in Cinema 4D

C4D: Cinema 4D

UI: User Interface

AI: Adobe Illustrator

AIn: Adobe InDesign

PS: Adobe Photoshop

AE: Adobe After Effects

AA: Adobe Acrobat

Introduction

Airway management encompasses the steps, techniques, and procedures used by physicians to maintain or alleviate airway obstruction. Airway management can be used in cases of emergency to remove a physical obstruction preventing the patient from breathing, or more commonly in cases where general anesthesia requires the patient to be mechanically ventilated via an endotracheal tube. The ultimate goal of airway management is to maintain oxygenation.

Intubation is the method of using mechanical ventilation to deliver oxygen to a patient through a tube placed in their trachea. Before surgery, the patient is anesthestized, the trachea is intubated and mechanical ventilation is used to maintain oxygenation and ventilation during the surgery. After the surgery is complete, the tube is removed (extubation) so that the patient can breathe on their own again. The risks of intubation are overall very low in an adult patient with normal airway anatomy.

When treating pediatric patients, it is especially important for the clinician to understand pediatric anatomy and to intubate cautiously and precisely. Pediatric airway structures are smaller and more easily damaged, and pediatric oxygen consumption is higher than adults, resulting in a higher risk of hypoxia. In cases of congenital disorders that directly affect the airway, such as micrognathia, accessing the airway becomes even more difficult. There is greater risk for complications if the physician is not aware of the techniques used in difficult pediatric airway management. Complications include injury to teeth, tongue, larynx, esophagus, and other soft structures of the oral airway (Whitlock 2017). In addition, if spontaneous ventilation is not maintained in the time before intubation is achieved, there is risk of hypoxia, subsequent brain and organ damage, or even death.

Micrognathia/Retrognathia

Micrognathia is defined as underdeveloped jaw, and retrognathia is defined as an abnormally posteriorly positioned jaw. Micrognathia may refer to a hypoplastic maxilla, however, in most cases the mandible will be the affected bone. Micrognathia can be associated with a number of congenital abnormalities. The three examined in this project are Pierre Robin sequence (PRS), Treacher Collins syndrome, and Goldenhar syndrome. This project specifically focuses on PRS, which is the most commonly seen and treated at Johns Hopkins Hospital. The PRS CT scans and laryngoscopy images/ videos were supplied by the Department of Otolaryngology at Johns Hopkins Hospital.

One of the most important issues surrounding micrognathia is the effect it has on the patency of the infant's airway. Because of the small jaw, the infant's tongue will often fall backwards, blocking inhaled

air from entering the larynx and reaching the lungs. Glossoptosis is defined as the downwards displacement of the tongue, and is a key marker of PRS. Micrognathic patients are typically labeled with a "difficult airway" upon birth and particular cautions are exercised during airway management. Generally, the assistance of an experienced anesthesiologist or otolaryngologist who has previously intubated pediatric difficult airways is highly recommended.

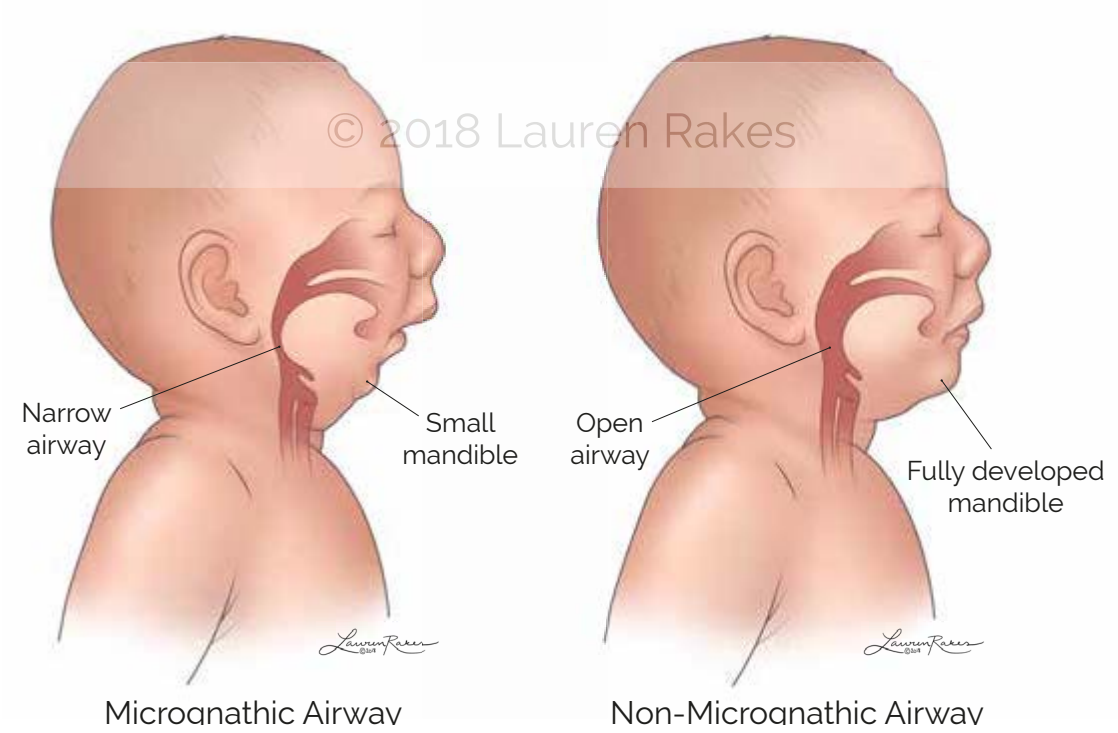


Figure 1. Micrognathic vs. non-micrognathic airway.

Pierre Robin sequence

Pierre Robin sequence (PRS) is a condition typically classified by a triad of abnormalities:

- Micrognathia
- Glossoptosis (posteriorly positioned tongue)
- Airway Obstruction

When these abnormalities present as an isolated condition in a newborn, they are diagnosed with nonsyndromic PRS. The triad can be a part of a wider range of abnormalities associated with a syndrome such as Treacher Collins syndrome or Goldenhar syndrome. In these situations, the child is diagnosed with syndromic PRS. Because of the variety in clinical presentation, the incidence of children born with PRS range from between 1:5,000 to 1:85,000 (Cladis et al. 2014).

Sometimes, a cleft palate is used to diagnose the syndrome, but it is not a necessary component of PRS. The cleft palate stems from the underdeveloped jaw. When the mandible does not grow appropriately during embryonic development, the tongue is displaced backward and upwards, blocking the growth of the soft palate. This blockage possibly leads to the U-shaped cleft palate seen in PRS patients.

The airway obstruction stemming from PRS can affect a baby's quality of life and ability to thrive almost immediately. Depending on the severity of the symptoms, the baby is put at risk if placed on its back or in any other position that may cause the tongue to block the airway. The hypoplastic mandible and cleft palate also make feeding difficult, thereby requiring that a gastric tube must be inserted.

There are a variety of preventative measures and surgical options that can improve the quality of life for PRS patients, such as tongue-lip adhesion, mandibular distraction, and tracheostomy. However, in some cases, the mandible may reach an appropriate size as the child grows without surgical intervention.

Treacher Collins Syndrome:

Treacher Collins Syndrome (mandibulofacial dysostosis) is a craniofacial condition presenting with a number of characteristics:

- Micrognathia/Retrognathia
- Downward-slanting eyes
- Eyelid coloboma (notch in lower eyelids)
- Underdeveloped/malformed ears
- Underdeveloped cheekbones/eye sockets/other facial bones

Like in PRS, Treacher Collins syndrome patients can present with a cleft palate. Other possible symptoms include hearing loss (caused by defects in the bones of the middle ear) and airway obstruction. The symptoms vary greatly in this condition, but the incidence of children born with Treacher Collins syndrome is thought to be 1:50,000 ("Treacher Collins Syndrome" 2018).

There are a variety of surgical options that can improve the quality of life for Treacher Collins patients. In addition, speech therapy and methods of improving bone conduction can be implemented in those affected by hearing loss.

Goldenhar Syndrome:

Goldenhar syndrome (facioauriculovertebral sequence) is included in a group of conditions that fall under the name of Craniofacial microsomia. These conditions can present with a large variety of possible skull and face abnormalities. It is unknown whether they are all the same condition with varying levels of

severity, or different conditions altogether. The specific symptoms for Goldenhar syndrome include:

- Facial asymmetry
- Underdevelopment of facial bones
- Eye abnormalities (ocular dermoid cysts, microphthalmia)
- Spinal abnormalities
- Ear abnormalities (defects in external ear, closed ear canal)

Other possible symptoms include micrognathia of the mandible or maxilla, airway obstruction, hydrocephalus, and other organ abnormalities. Symptoms can vary in severity and in type. Like children with PRS or Treacher Collins, patients with Goldenhar syndrome may have trouble feeding and breathing. The incidence of children born with Goldenhar syndrome, because the condition is not well defined, is thought to be between 1:3,500 to 1:25,000 ("Goldenhar Disease" 2017).

Difficult Airway

A difficult airway is defined as "...the clinical situation in which a conventionally trained anesthesiologist experiences difficulty with facemask ventilation of the upper airway, difficulty with tracheal intubation, or both" (Apfelbaum et al. 2013). A variety of factors can influence the difficulty of an airway, including congenital defects, inadequate seal of devices, airway obstruction, or injury to the airways. A difficult airway can be dangerous and can lead to injury when the patient needs to be intubated quickly, or when repeated intubation attempts are made. Before a surgery or anesthetic care, the patient is always evaluated for risk of a difficult airway by researching the patient history and performing a physical exam.

For newborns with micrognathia, it is important for the physician to have an airway management plan available immediately upon birth in case the airway is so severely obstructed that the child cannot breathe. In many cases, newborns with micrognathia undergo a variety of surgical procedures soon after birth to help alleviate the obstruction. In these surgical cases, intubation preceding the surgery is necessary and the airway is often very difficult to access.

Difficult Airway Algorithm

The "Difficult Airway Algorithm" created by The American Society of Anesthesiologists (Figure 2) was designed to help physicians make the most appropriate decisions during adult tracheal intubation so as to avoid unnecessary risk and injury to the patient. The algorithm is presented as a black and white flowchart. While key principles of the original difficult airway algorithm may also apply to pediatric

cases, there are some recommended techniques and equipment that are better suited for adult patients and could even cause harm if used on a pediatric patient. In the case of infants (particularly micrognathic or retrognathic patients) the airway algorithm needs adaptation to accommodate the physical and developmental needs of the age group and conditions.

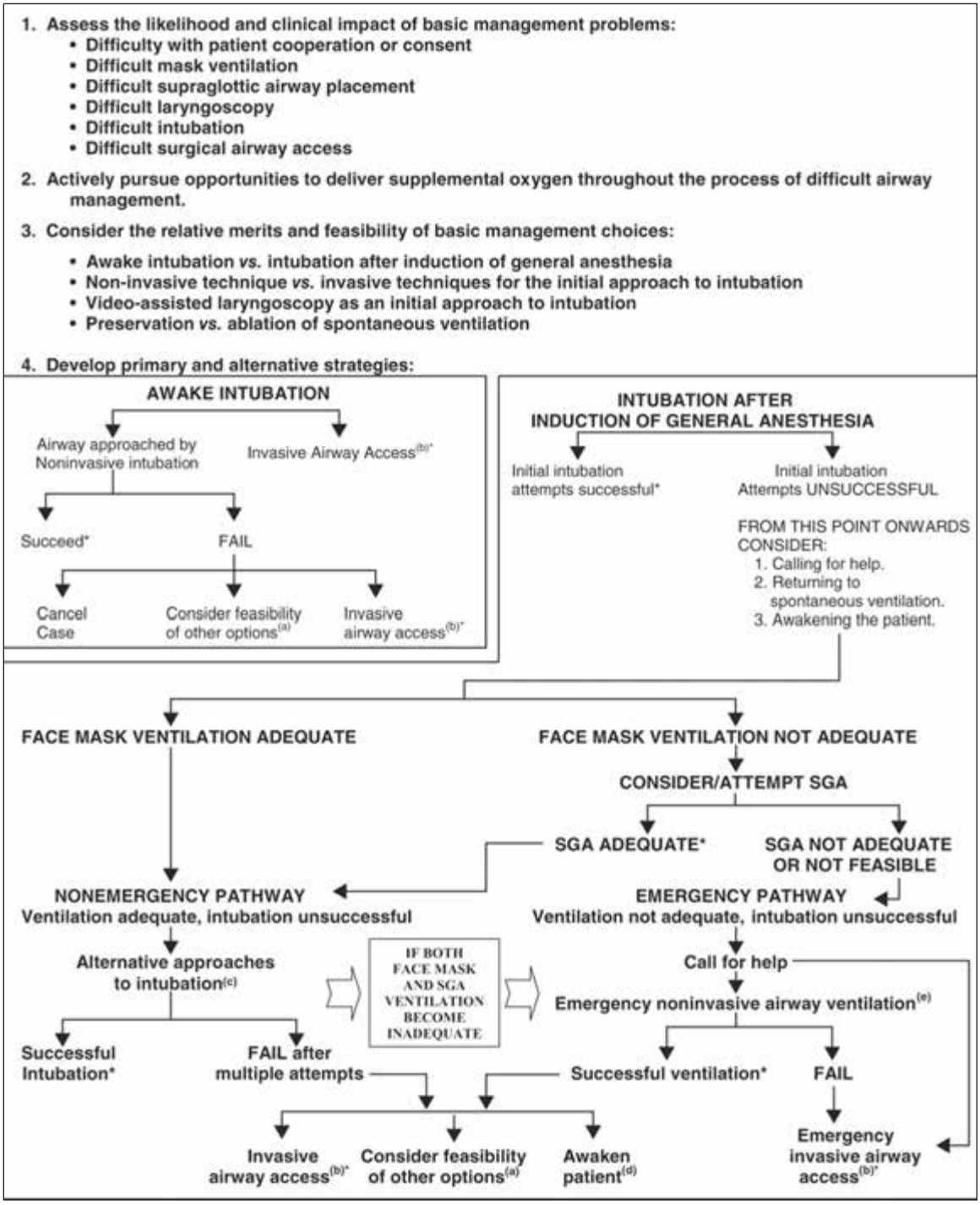


Figure 2. ASoA's Difficult Airway Algorithm (Apfelbaum, Jeffrey L. et al. 2018) *(text not intended to be read).*

Current Teaching Aids

Anatomical model mannequins are commonly used in the education of anesthesiologists to train for airway management and intubation. The mannequins typically consist of a plastic life-like head and shoulders with a plastic tongue, oropharynx, epiglottis, larynx, vocal cords, and trachea. Often the trachea are attached to inflatable lungs. Training physicians can use laryngoscopes and other equipment to practice airway management directly on the mannequin. There are models that can adjust to account for a difficult adult airway, as well as separate models specifically designed for difficult airways.

The pitfalls to teaching difficult airway access on micrognathic infants using mannequins are the limits to the realism (lack of secretions, anatomic variations, and physiologic decompensation), high cost, and number of available products. Currently, the only Pierre Robin sequence pediatric mannequin commercially sold is created by AirSim (Figure 3). The model has mandibular hypoplasia, glossoptosis, and a cleft palate. Users can practice nasal and tracheal intubation, as well as the insertion of supraglottic devices (nasopharyngeal airway, oropharyngeal airway, or laryngeal mask airway). While this simulation tool is intended to help new and practicing anesthesiologists refine their technique, the mannequin is not widely accessible. In hospitals with limited resources, it may be cost prohibitive. In addition, because Pierre Robin sequence is so rare, hospitals may place this simulator low on their list of priorities.

Figure 3. AirSim Pierre Robin sequence infant training mannequin ("Airway Management Training Manikin..." 2018).

A smartphone app called “The Difficult Airway App” intends to help users master airway management skills (Figure 4). The application contains a section with mnemonics for memorizing how to predict a difficult airway, a section for airway algorithms, a section on Succinylcholine hyperkalemia (an anesthetic drug), a rapid sequence intubation calculator for adult and pediatric patients, a section for tips and tricks, and an additional educational resources section. The algorithm section of the application shows only a screenshot of an algorithm displayed in the textbook Rosen’s Emergency Medicine, by Ron Walls, M.D. A recent update contains a modified algorithm with “predicted difficult airway and experienced difficult airway situations with modifications to account for transport times, local protocols, preference and medical direction” (Airway Management Education Center 2018).

The app is beneficial for training physicians because of its accessibility and portability. When used in combination with other medical education tools, this application may help improve the recall of a training anesthesiologist. While educational, this app lacks interactivity within its algorithm section. It also only covers difficult airway techniques that are typical for adult patients, even though its Rapid Sequence Intubation section has a pediatric patient option.

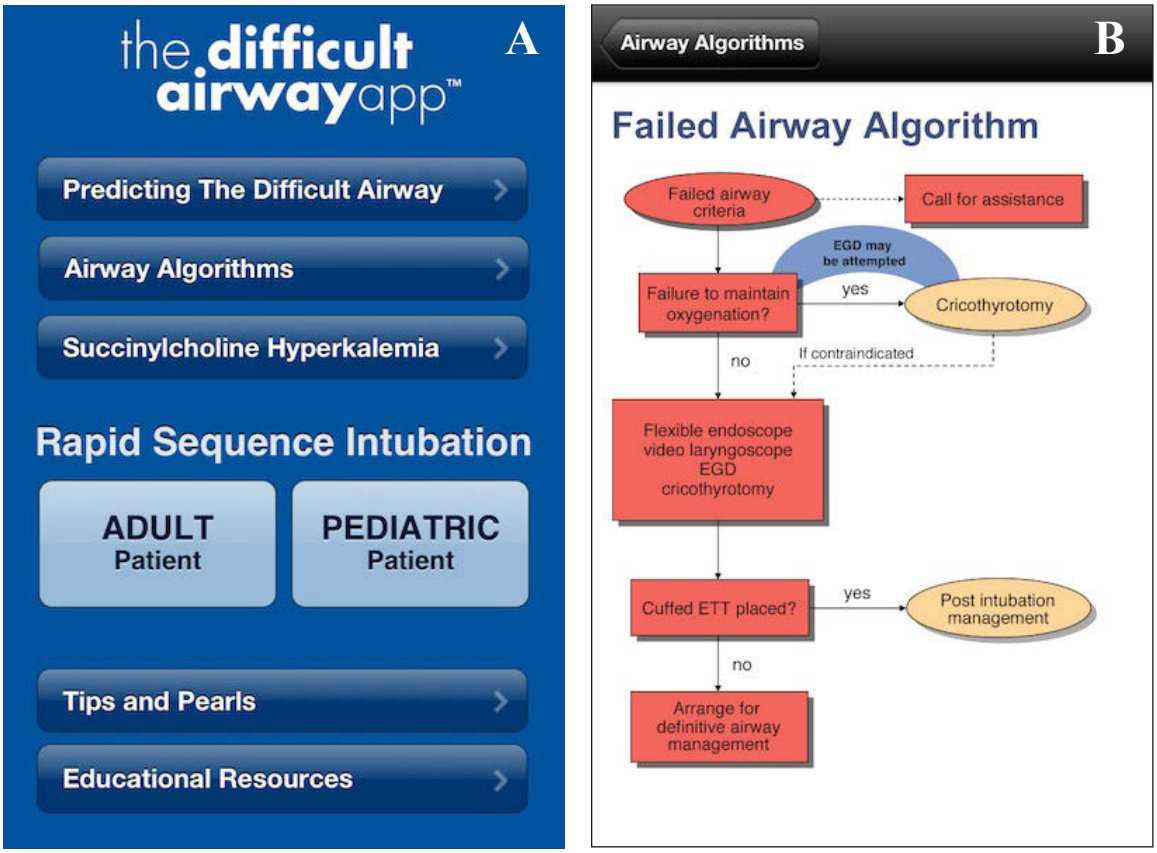


Figure 4. The Difficult Airway App (Airway Management Education Center 2018). **(A)** Title screen. **(B)** Airway Algorithm screen. *(Text in screenshot not intended to be read.)*

Another resource available is the Airway Ex app for smartphones (Figure 5). This app serves as a virtual intubation trainer. The user can go through a simulation of intubation by tapping and sliding on their smartphone screens. Several factors affect the user's score, such as the amount of secretion in the field of view, the amount of injury to the tissue, and the time in which the intubation attempt is completed. The benefits of using this application include being able to practice with a variety of virtual intubation devices on multiple unique cases. The user is able to view a highly realistic airway that reacts to contact from the virtual devices.

This application is an excellent tool for the training anesthesiologists, certified nurse anesthetists and other physicians. Presented in game format, users are able to earn continuing education credits, help their recall, and possibly improve their intubation skills. Currently, the app features four cases of pediatric intubation, including two difficult airway scenarios (angioedema and laryngomalacia). The app could be improved or expanded with the addition of more cases of infant difficult airways, especially one exhibiting micrognathia. If these cases were included, it could be cost-effective way to supplement the training of new pediatric anesthesiologists and otolaryngologists.

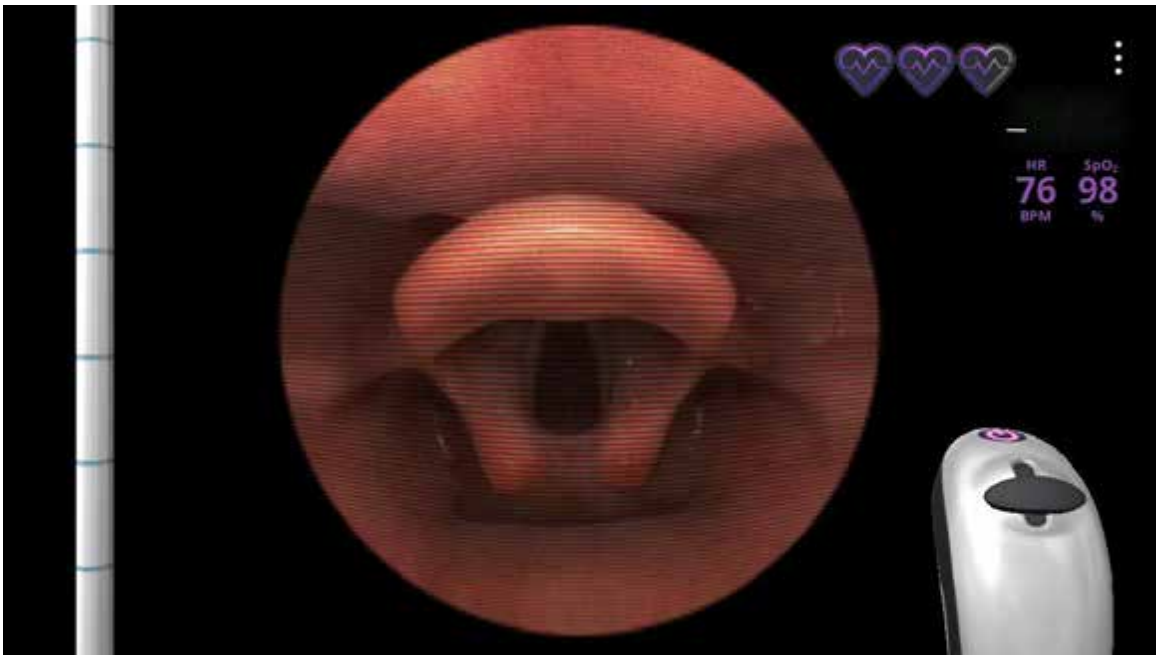


Figure 5. Airway Ex app (Level Ex, Inc. 2018).

Despite the effectiveness of the Airway Ex app, it is crucial that physicians use other tools to practice and build their airway management skills. The cases available for simulation in the application are heavily controlled. The app gives specific directions about what to do for each case, but the user can

only manipulate the laryngoscopy device and the endotracheal tube. They have no control over patient positioning or any other aspect of airway management. In real intubation scenarios, the user will have to make quick unanticipated decisions. Because of this, it is important that they are familiar with all the available options.

The application created for this research project aims to fill the gap in the educational tools currently on the market and provide users an accessible way to learn about micrognathic infants using an algorithm created specifically to help manage their airways.

Algorithms as Learning Tools

An algorithm is defined as a set of rules for solving a problem. In medicine, algorithms are used as training tools. The physician follows the steps along the algorithm to prepare for possible outcomes in a scenario. The goal of an algorithm is to build decision-making skills, which is especially important for new or training doctors. Algorithms are widely accepted and used in a variety of different medical specialties outside of anesthesia, including first aid, pain management, and emergency medicine (Schwarz et al. 2013).

The use of algorithms as teaching tools, especially within the field of medicine, is not without its critiques. One concern is that the user may choose to follow the algorithm without understanding the "reasoning and principles upon which the algorithms are based (AJPH 1982). For this reason, the algorithm created for this project includes multimedia and text explanations for each step of the process. Algorithms should not be used as a rigid rule book for how a procedure should be performed, instead, the input and logic of the physician trained on an algorithm should be the most important factor.

The algorithm created for this web application allows the users to choose to be guided through the algorithm or to explore freely. The users can jump to any point of the algorithm to explore a specific topic.

Use of Interactive Media

Interactive media is an effective method of learning new information and supplementing existing knowledge. Interactive media includes any media that uses a digital environment to present information to the user in response to the user's actions. By including interactivity, an application or module relies on the user to progress. Interactive media is increasingly being incorporated into classrooms of every kind, and helps the users learn and retain more information through experimentation (Lingnau et al. 2003). Interactivity is utilized in all aspects of this application, including manipulation of the 3D model and exploration of the algorithm.

Learning Theories

During the planning of this project, various learning theories were examined and considered for implementation in the web application. Elements from multiple learning theories were ultimately used to ensure the best knowledge retention.

The main learning theory implemented was Malcolm Knowles' theory of Andragogy. At its core, andragogy is used to direct adult learning by independent exploration and involvement. In order to get the most out of the web application created for this project, the users must direct their learning and choose to explore. One principle of andragogy as a theory of learning is that "Adults are most interested in learning subjects that have immediate relevance to their job or personal life" (Culatta 2015). Because this application is made for practicing and training pediatric anesthesiologists and otolaryngologists, the knowledge within the application is directly relevant to their careers and goals. If the physicians are eager to learn how to improve an aspect of their job, they are more likely to take the initiative to do so.

Another learning theory implemented is Carl Rogers theory of Experiential Learning. Under this theory, users are more likely to retain information and learn more effectively when external threats are at a minimum. Reviewing this application in their spare time and outside of the operating room ensures that the learning takes place at a time where the physicians are not in danger of making mistakes and endangering a patient. Practicing and learning outside the clinical environment ensures that when the time comes to perform the actions on an actual patient, the clinicians will feel more confident and well prepared.

The final learning theory used is J. Carroll's theory of minimalism. While this application is text and image heavy - usually an antithesis to minimalism, some other aspects of the learning theory apply. One principle of minimalism is to "Make all learning activities self-contained and independent of sequence" (Culatta 2015). By breaking up the three sections of the application (3D Anatomy, Difficult Airway Algorithm, Background Information) the users can choose which learning activity to engage with at any time. Additionally, the application assumes the users have a general knowledge of anesthesia from medical school or other training. By leaving out specifics (such as the definition of an endotracheal tube) the app can focus on important, new information that will be useful for training clinicians.

WebGL

WebGL (Web Graphics Library) was chosen as the medium for this application because it increases

the accessibility as compared to other methods. WebGL applications can be run on any computer without additional software. WebGL applications can use any modern browser (Google Chrome, Mozilla Firefox, Safari, Internet Explorer, Opera) and allows for the manipulation of 3D models without the use of Adobe Flash.

Objectives

i. Construct an application for learning and reviewing difficult airway management in infants with micrognathia/retrognathia, for distribution to pediatric otolaryngologists and anesthesiologists within Johns Hopkins Hospital (and eventually to hospitals nationally and internationally).

ii. Improve knowledge of micrognathia anatomy in infants through the creation of an interactive 3D model of a Pierre Robin sequence infant (rotatable, on/off for certain anatomical features, sagittal cross-sections).

iii. Increase familiarity with a pediatric difficult airway algorithm specific to micrognathia by way of self-guidance through an interactive decision-making flowchart.

iv. Describe steps in the flowchart with illustrations, 2D animations, and supporting text.

v. Create a PDF resource of the information available in the app, to be presented online and in a printed full-color booklet.

vi. Create a foundation and workflow for expanding upon the interactive flowchart methodology with other airway management scenarios or different procedures.

Audience

The primary audiences of this WebGL application are pediatric anesthesiology and otolaryngology residents and fellows, and certified registered nurse anesthetists who anticipate that they will need to manage a pediatric micrognathic difficult airway in a clinical setting.

Materials and Methods

Workflow

A goal of this project was to create a workflow that can be used as a guide for the creation of future medical educational applications. Other medical specialties, such as critical care medicine, would benefit from a similar application with interactive algorithms.

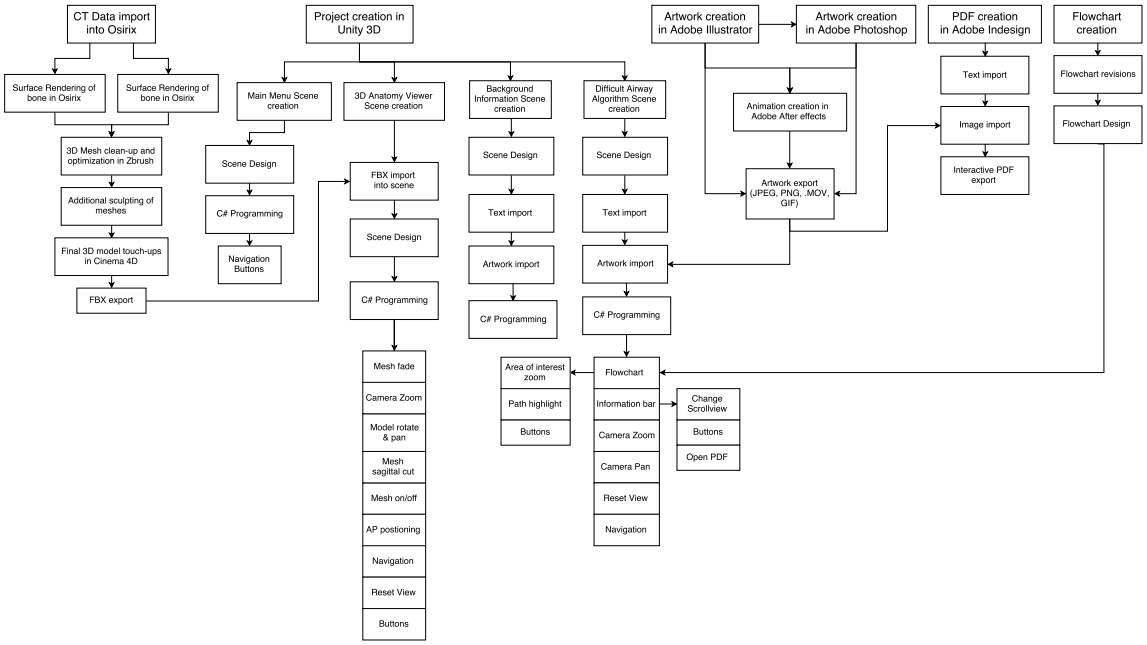


Figure 6. Overall project workflow *(legible text on subsequent images).*

The entire workflow (Figure 6) can be subdivided into smaller sections:

i. Processing CT data of a patient for inclusion within a Unity-built application (Figure 7 (A)).

ii. Design of an interactive flowchart: Research and creation of flowchart, mapping of the flowchart for usability, design, and then programming the flowchart within an application to simulate real-life decision-making (Figure 7 (B)).

iii. Creation of artwork and assets, including linework in illustrator, rendering in Photoshop, and animation with after effects (Figure 7 (C)).

iv. Creation of an online interactive PDF of text and figures (Figure 7 (D)).

v. Building and programming a WebGL application within Unity 3D, with both 3D and 2D scenes, that can be hosted on the web for ease of access (Figure 8).

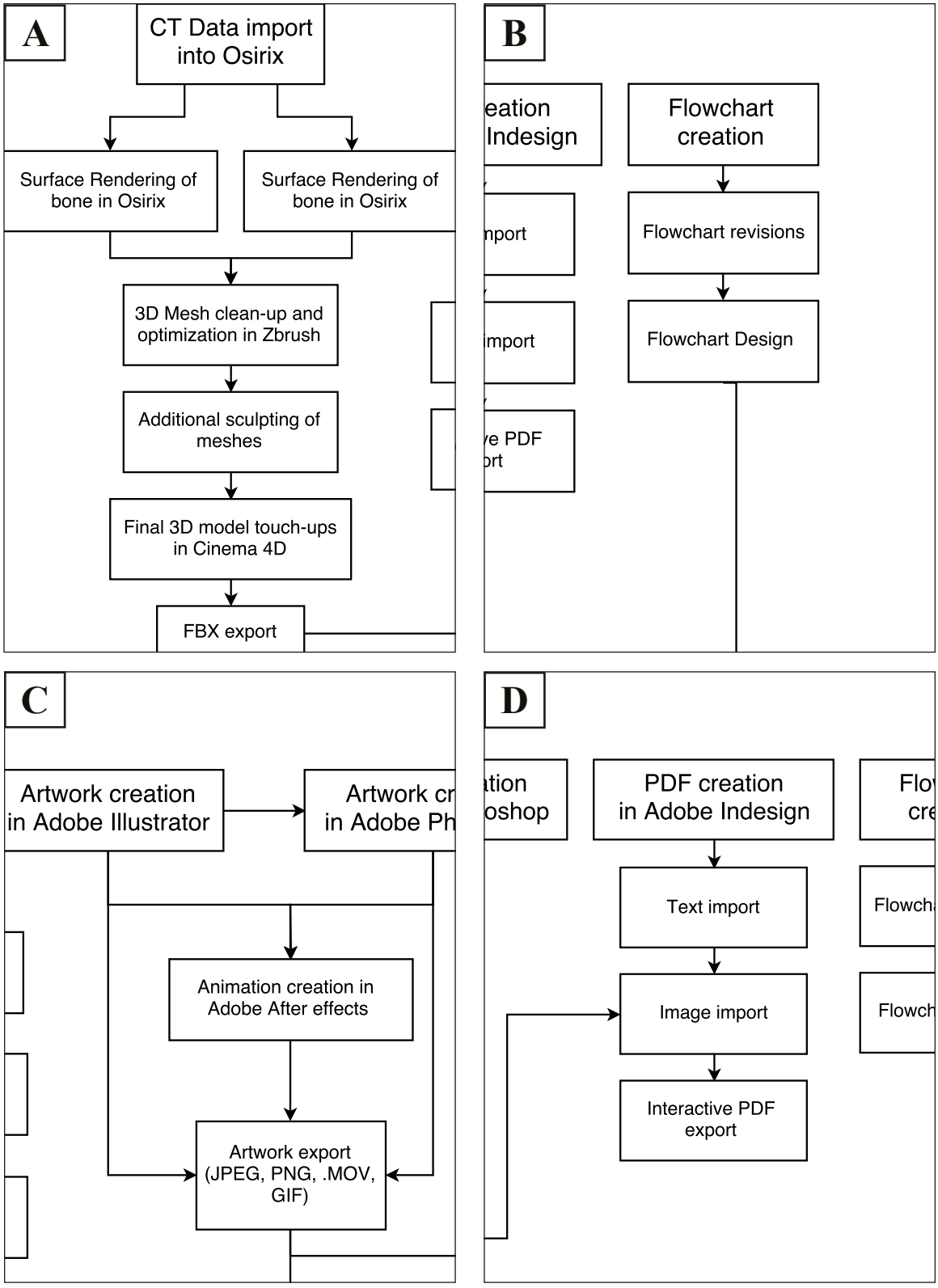


Figure 7. Details of overall project workflow. **(A)** CT data import. **(B)** Flowchart creation. **(C)** Artwork creation. **(D)** PDF creation.

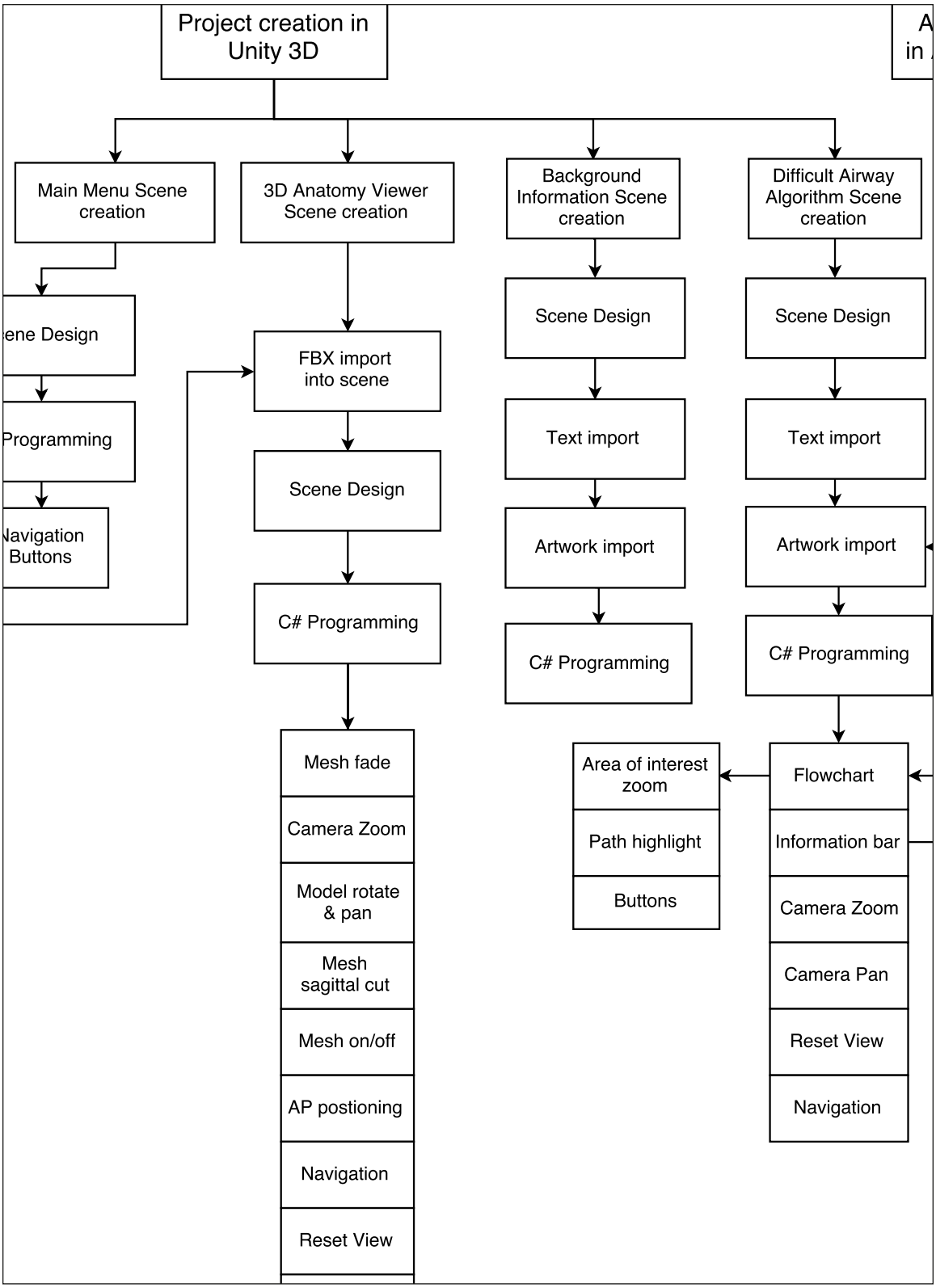


Figure 8. Details of overall project workflow: Unity 3D workflow.

Software and Equipment

A range of software was used to create the final project and all of its assets (Table 1). OsiriX was used to extract OBJ 3D models of bone and skin from the DICOM (Digital Imaging and Communications in Medicine) data set of a Pierre Robin sequence patient. Pixologic Zbrush was used to clean up the OBJ files exported from Osirix, and also to sculpt additional features of the 3D model such as the larynx, tongue and hyoid bone. Cinema 4D was used to refine the final 3D sculpt and render images of the model with materials and lighting. Unity 3D was used to build and program the final WebGL application. The script used in Unity was written in MonoDevelop. Adobe Illustrator was used to create the lineart that served as the basis for the artwork throughout the application, as well as for labeling of the final illustrations. Adobe Photoshop was used to render artwork using line art from Adobe Illustrator. Adobe After Effects was used to create 2D animations from the artwork created in the other programs.

Software	Equipment
OsiriX	Wacom Intuos Pro tablet
Pixologic Zbrush	iMac 27-inch
Maxon Cinema 4D	iPad Pro 12.9”
Unity 2017	
MonoDevelop Version 5.9.6	
Adobe Illustrator	
Adobe Photoshop	
Adobe After Effects	
Adobe InDesign	
Adobe Acrobat	

Table 1. Software and equipment.

Data Acquisition

One CT scan of a Pierre Robin sequence infant was completed before the start of this project, and the resulting anonymized DICOM file was obtained from the Department of Otolaryngology and Head and Neck Surgery at Johns Hopkins Hospital with the help of Dr. Jonathan Walsh. Images and video of laryngoscopy performed on multiple micrognathic and retrognathic patients were also provided. An opportunity was available to view and sketch the intubation of a Pierre Robin sequence infant in the

operating room, which helped in creating the illustrations.

Surface Rendering in Osirix

The CT DICOM file was imported into OsiriX. A series of 237 images at 0.6 mm intervals was chosen because of the large number of slices and the clarity of the structures. In order to segment out different anatomical structures within the file, a **Surface Render** was created (3D Viewer > Surface Render) (Figure 9). **Decimate-Resolution** was set at 0.5. **Smooth-Iterations** was set at 20. Experimentation with **Pixel Value** yielded -500 as the best setting for a skin surface render, and 100 best for the skeleton. This created a model visible in the OsiriX viewer for each render (Figure 10 and Figure 11). The model can be rotated and examined in OsiriX, and the surface render can be adjusted to fill in any gaps or create a more detailed model. The models were exported as **.OBJ** files (Export 3D-SF > Export as Wavefront (.OBJ)). Resolution of the OBJs was kept low, as they would only be used as a template for sculpting in Zbrush.

Figure 9. Surface render dialogue box.

Model Creation in Zbrush

The following workflow in Zbrush was used to create the master 3D model used in the application.

Import and Cleanup

First the skeleton OBJ file was imported into Zbrush as a **SubTool** (Figure 12). The mesh was rough

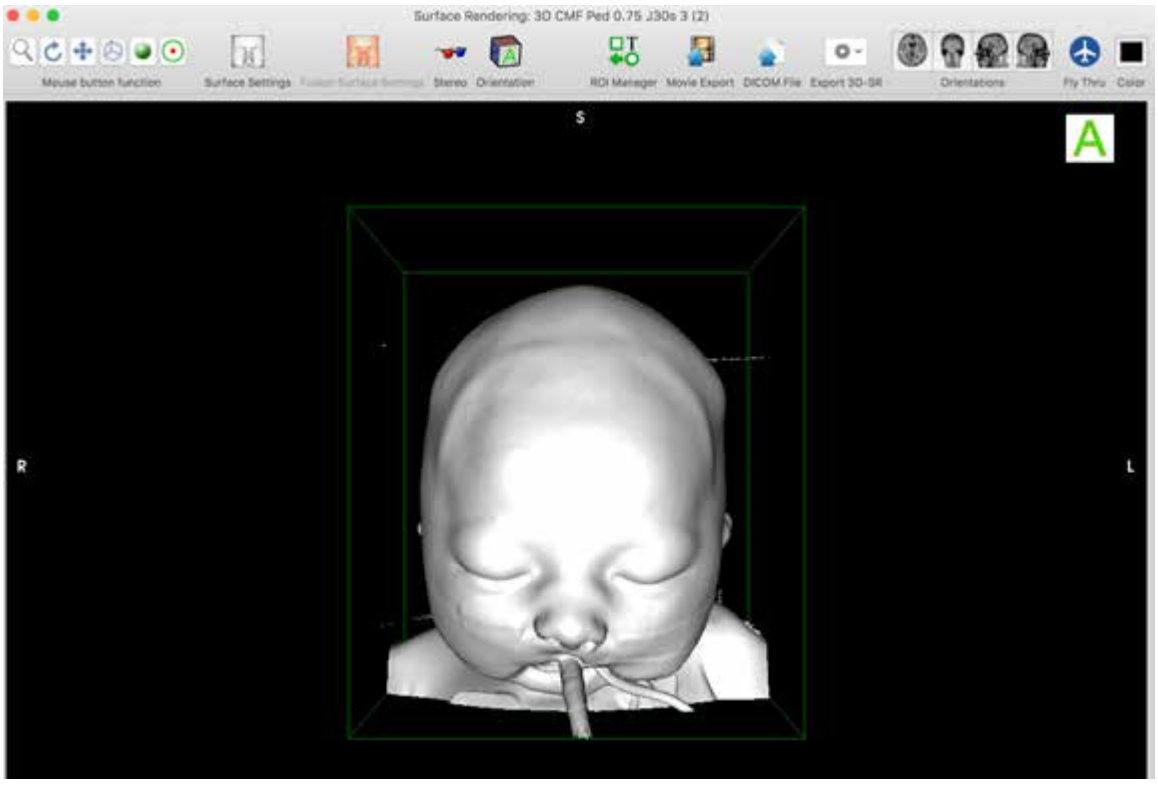


Figure 10. Skin surface render *(text not intended to be read).*

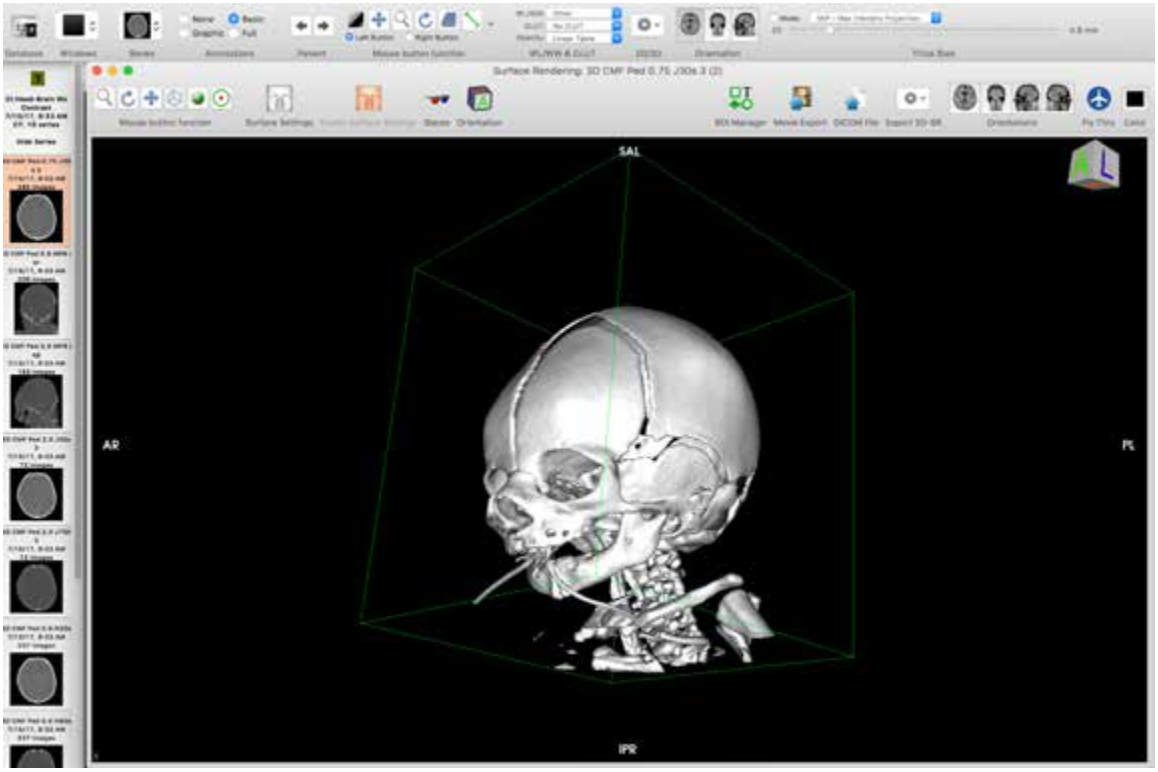


Figure 11. Skeleton surface render *(text not intended to be read).*

and included extra equipment captured in the CT scan that was not necessary for the final model. The SubTool was cleaned up in Zbrush using the **Smooth** brush (Any brush + Shift Key). Extraneous pieces of mesh were masked out with the **SelectRect** tool (Shift + Option + Command + Left click and drag). For better control over which pieces of mesh to delete, the **Select Lasso** was used (Brush panel > Lasso Select). Once undesired pieces of mesh were masked out, they were deleted (Geometry > Modify Topology > Del Hidden).

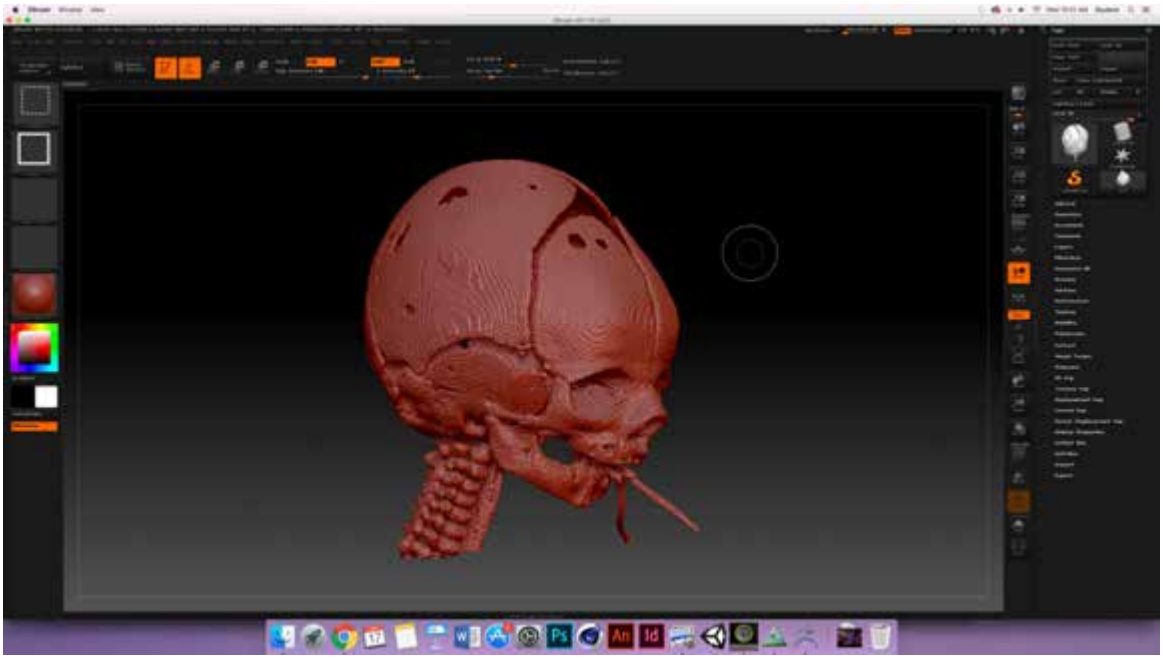


Figure 12. Skeleton model in Zbrush *(text not intended to be read).*

Once the SubTool had all extraneous meshes removed, defects in the scan were corrected using brush tools. The **ClayBuildup** brush (Brush panel > ClayBuildup) was used to fill in large hollows in the SubTool. Paired with the smooth brush, this allowed for quick natural sculpting. **Damien Standard** was used often (Brush panel > Dam_Standard). This brush creates a thin valley in the mesh, useful for sculpting parts of the SubTool that required divots and holes. The **Move** brush (Brush panel > Move) was used to select a section of the SubTool and move it around depending on which way the model is facing. By holding the **option** button, it can be constricted to “in” or “out” movement of the mesh.

During the process of cleaning and refining the SubTool, **DynaMesh** (Geometry > DynaMesh > DynaMesh Button) was used to retopologize the mesh (Figure 13). **Blur** was set to 0, **Resolution** changed depending on what stage the sculpt was in, and **Polish** was usually turned off. DynaMesh is a particularly useful feature of Zbrush. It allows for adjustment in the level of subdivisions in a SubTool. If part of the

model is stretched, the vertices usually get pulled and separated so that the polygons between them create disrupting facets in the SubTool. By clicking DynaMesh, the vertices are redistributed along the mesh, resulting in a much smoother surface. Dynamesh is essential, especially when sculpting a new object from a sphere or a cube.

On occasion, holes would appear in the mesh, either as a result of sculpting or from the original imported file. These were easily fixed by selecting the Move brush, overlapping pieces of the surrounding mesh overtop of the targeted hole, and using DynaMesh. The two overlapping pieces melded together, making a single mesh without a hole. Once the geometry was complete, the Smooth brush was used to erase any evidence of the hole (Figure 14).

Figure 13. Dynamesh.

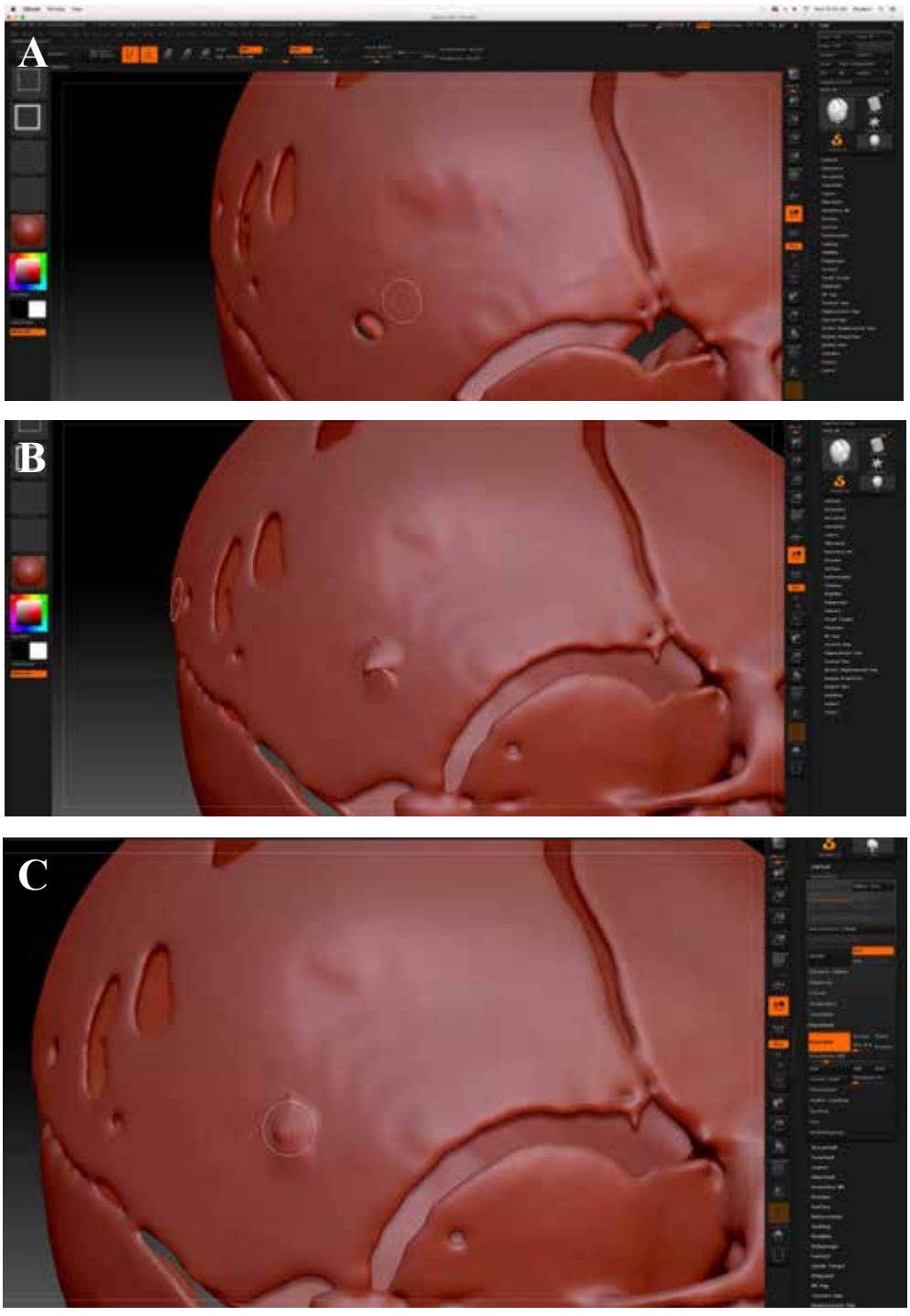


Figure 14. Closing holes in the mesh. **(A)** Finding a hole. **(B)** Overlapping the mesh. **(C)** Hole closed. *(Text not intended to be read.)*

Separating Polygroups

The skeleton SubTool had to be split into separate pieces: the skull, the mandible, the spine, and other extraneous pieces like the ribs and clavicle. This was necessary in order to turn on and off the visibility of these elements separately in the final application. To split the SubTool, the different pieces were first separated into **polygroups**. There are a variety of ways to create polygroups in Zbrush. A masking method was chosen for this project. First, the object that was to be separated was masked out using either **MaskRect** (Command + Left Click Drag) or **MaskLasso** (Command > Brush Palette > MaskLasso). Once masked, the **Group Masked** button was pressed (Polygroups > Group Masked) (Figure 15). This created two polygroups: one of the original mesh without the targeted object, and one polygroup that holds only the targeted object. To view the colors of each polygroup, the **Line Fill PolyF** button was clicked. After, the **Groups Split** button was selected (SubTool Panel > Split > Groups Split). This action split the two polygroups into separate SubTools. There were holes in each SubTool after separating them, and these holes were closed by selecting the subtool, and clicking **Close Holes** (Subtool Panel > Modify Topology > Close Holes). These actions were repeated for each separate piece extracted from the original SubTool.

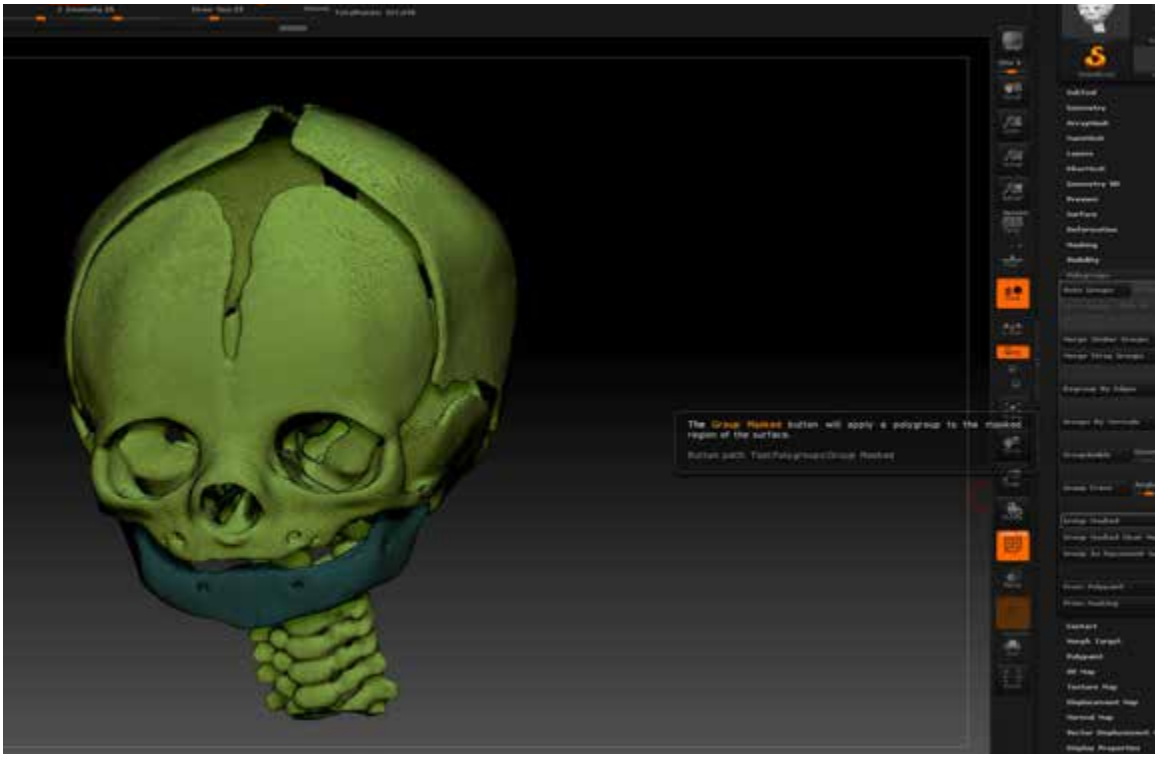


Figure 15. Polygrouping *(text not intended to be read).*

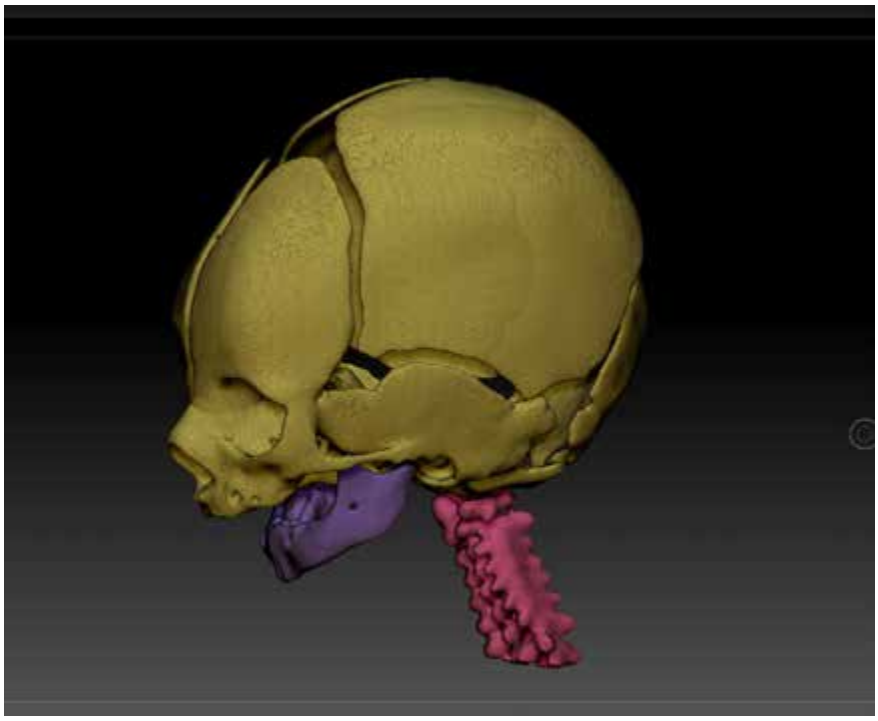


Figure 16. Polygroups.

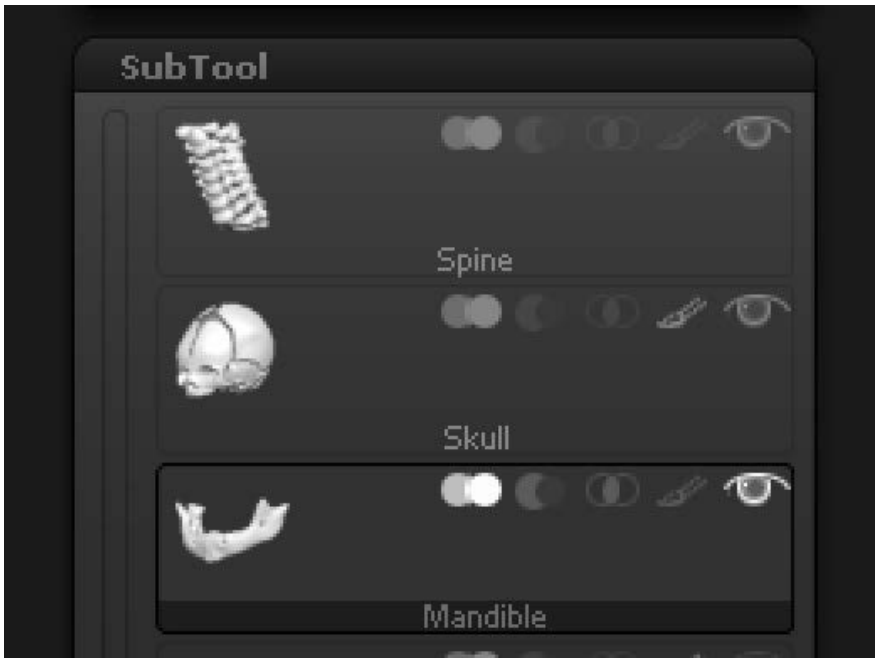


Figure 17. Separated subtools.

The process mentioned above was used to refine the skin SubTool as well. The face was made more symmetrical by using the Move tool and sculpting the face with various brushes. The patient's ears had been flattened by the restraint during the CT scan, so a new ear was sculpted. The polygrouping method used above was also used to separate the new sculpted ear from the rest of the mesh. The new ear SubTool was duplicated within the SubTool menu. The duplicate SubTool was moved to the other side of the model with the **Transpose** tool (Brush Panel > Transpose). The ear was flipped to right orientation by mirroring it across the x-axis (Deformation > Mirror (x symbol highlighted)). The ear was transposed by clicking in the center of the ear, and dragging out an axis with the Transpose tool. Clicking and dragging on the middle circle of the Transpose tool dragged the entire ear mesh along the Transpose axis (Figure 18).

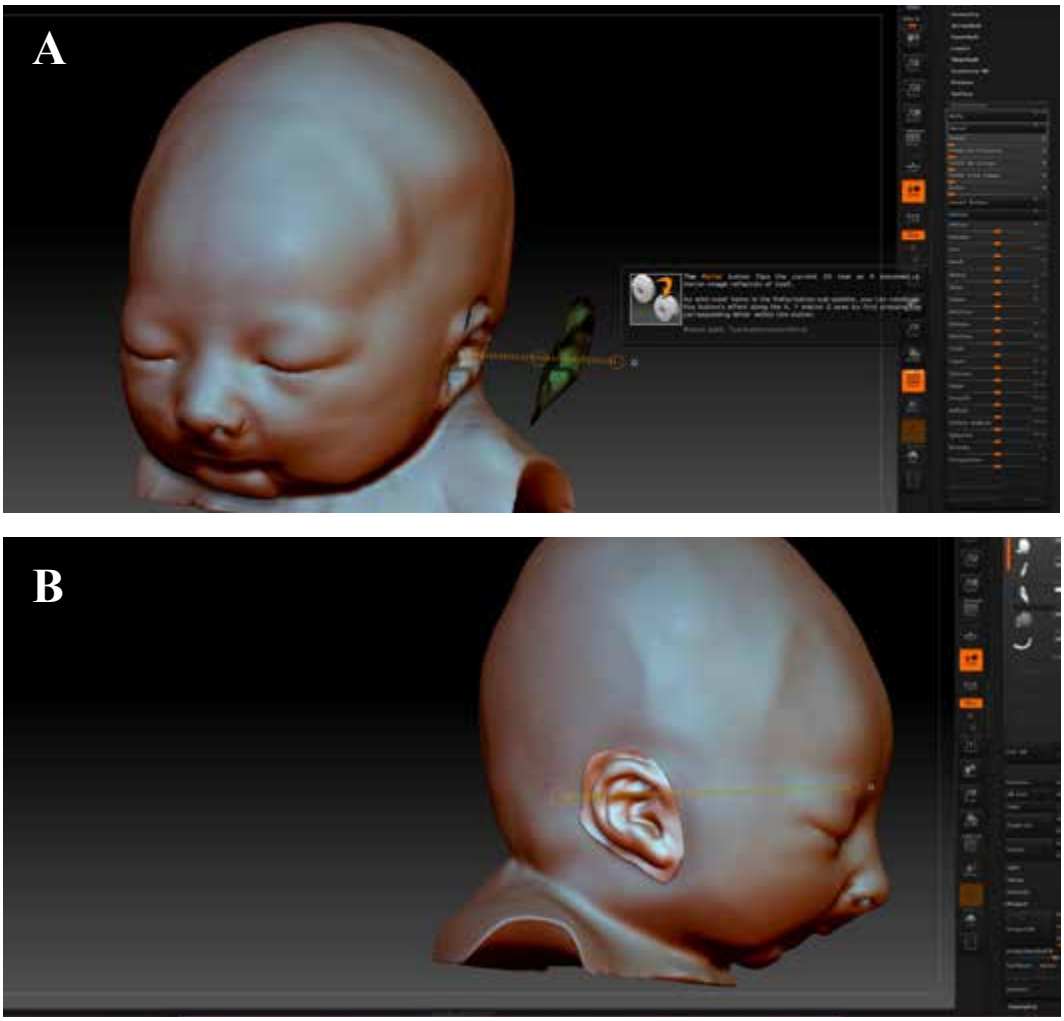


Figure 18. Duplicating the ear. **(A)** Subtool duplication. **(B)** Transposing ear. (*Text not intended to be read.)*

Once the new ear was in place, it was placed above the Skin Subtool in the menu (Subtool > Right Crooking Arrow) and a Merge Down action was performed (SubTool > Merge > MergeDown). This unified the new ear to the other side of the skin mesh.

Some structures that were necessary to include in the final interactive application were quicker to sculpt in Zbrush than attempting to isolate the structures in the CT DICOM data set in OsiriX. These structures included the larynx, the tongue and mouth floor muscles, and the hyoid bone (Figure 19). For each piece, a new SubTool was inserted into the project (usually a plain sphere). Then, using the move tool and various brushes, the SubTool was sculpted into the right anatomical shape. Cross-sections from the CT data viewed in OsiriX were used as reference for size and shape of the new structures.

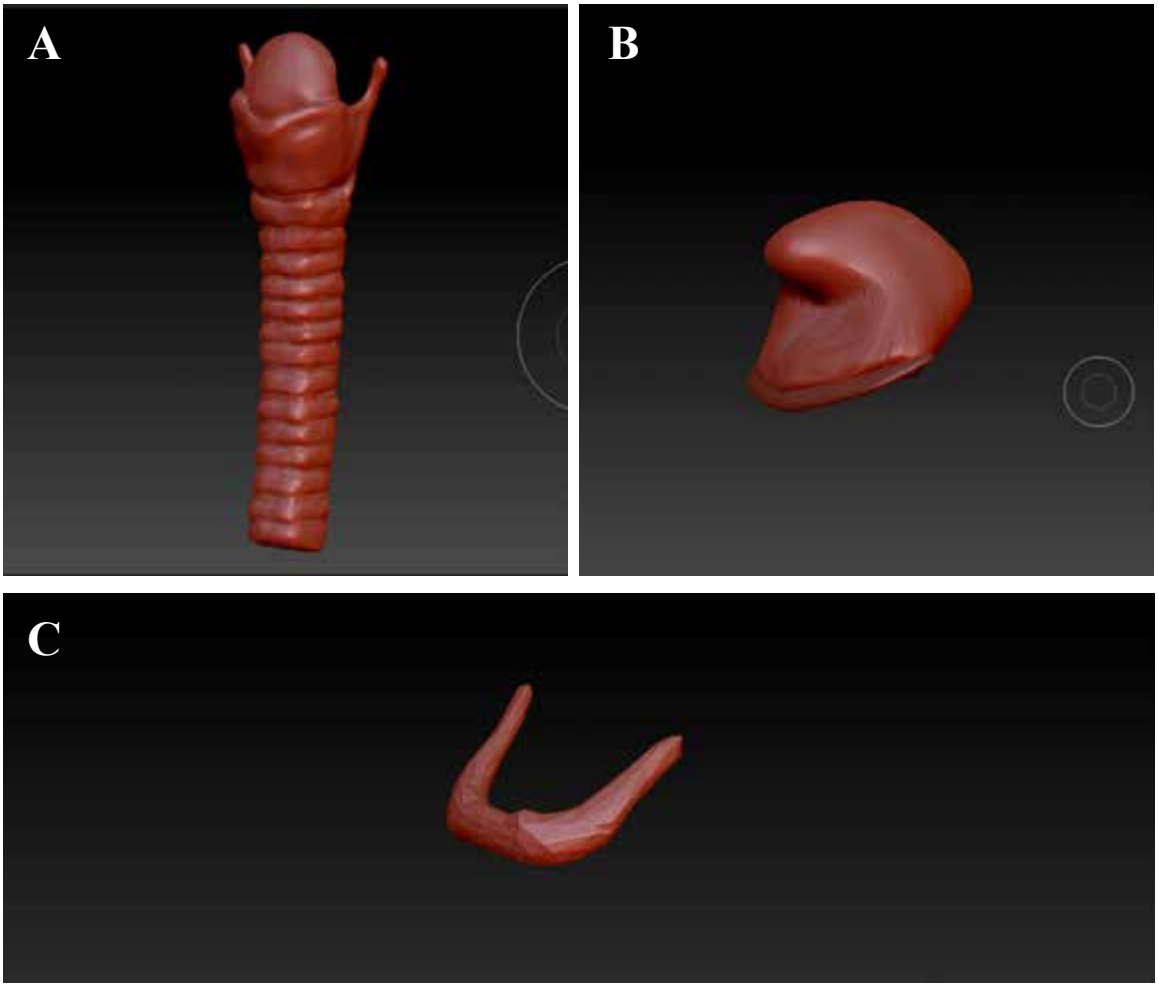


Figure 19. Sculpted subtools. **(A)** Larynx. **(B)** Tongue and mouth floor muscles. **(C)** Hyoid bone.

Once all the necessary SubTools were sculpted, each one was duplicated and cut in half to create a sagittal cross-section version. In the final application, the user can click a “Sagittal” button and view only half of the 3D model to better visualize the internal anatomy. The sagittal cut was performed in Zbrush by rotating the entire model until it was front facing, selecting each duplicate SubTool, and using MaskRect and **Del Hidden** to delete the left half of each mesh. Holes left behind were filled with **Close Gaps**. In the sagittal cross-section of the skin, the nasopharynx, oropharynx, soft palate, and esophagus had to be sculpted into the new flat medial surface (Figure 20). The ClayBuildup brush was used primarily for sculpting these airways. By holding Alt, the brush is reversed, allowing scraping into the mesh as opposed to adding onto it. This was useful for creating impressions onto a flat surface.

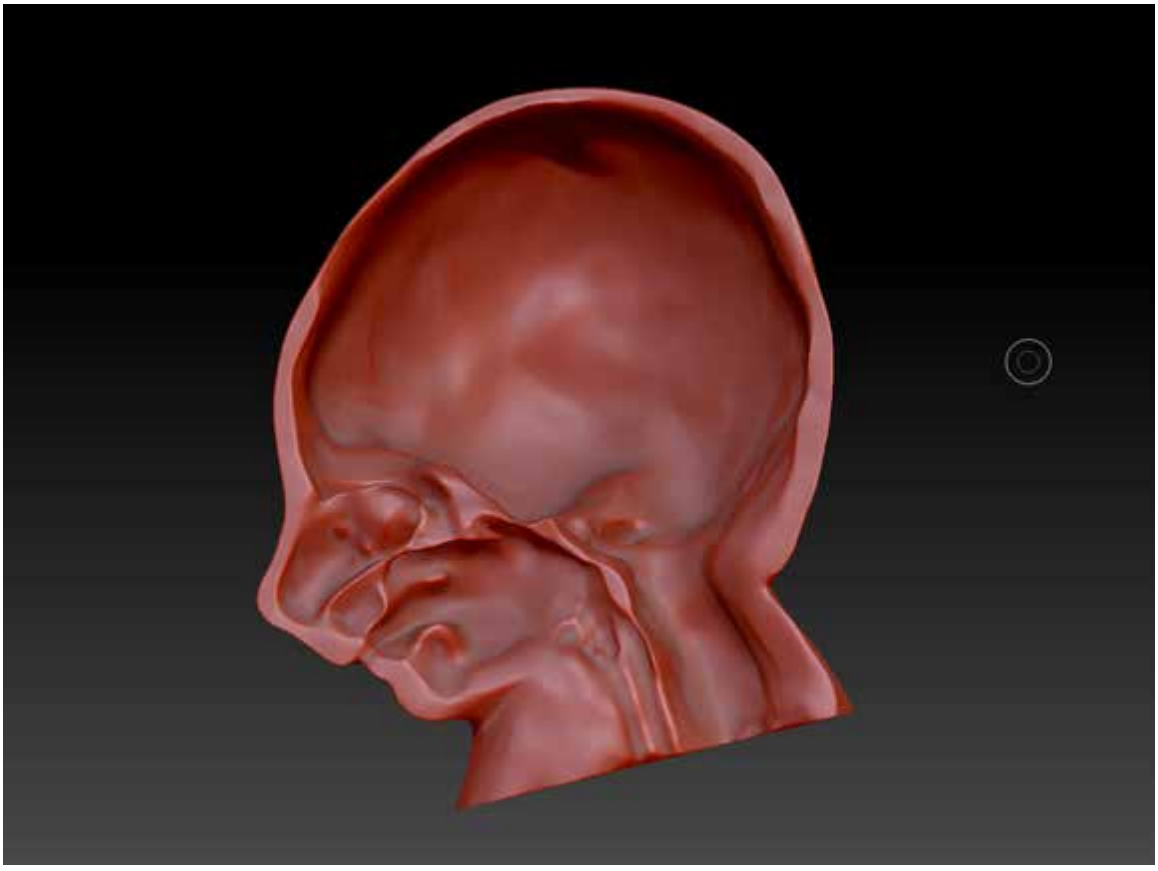


Figure 20. Sagittal skin.

Finishing Model in C4D

After sculpting in Zbrush, the model’s base was uneven. A flat base was needed for the final application. Boolean operations are tedious in Zbrush, especially when the operation needs to be done on

multiple SubTools at once. The decision was made to use C4D to do the final subtractive boolean operation on the model.

Each SubTool of the model was exported by using the **GoZ** plugin (Tool panel > GoZ) (Figure 21). When clicked, this plugin exports the selected SubTool and opens it up in a document in C4D. Each subtool was exported into the same C4D document, with the correct alignment and registration. Once in C4D, the entire model or its separate components can be manipulated like in any other C4D project.

Figure 21. GoZ plugin *(text not intended to be read).*

To make the flat base, a **Boole** (boolean operation) object was created to contain all the model's separate objects in addition to a large cube. To start, an empty **null** was created to hold all the model's objects (Select all layers + Option + G). A **cube** object was created, and placed at the appropriate angle with the mesh so that a resulting subtractive boolean operation would create a flat bottom (Figure 23). A Boole was created (Array button > Boole) and placed at the top of the layer panel. The null containing the objects was placed in the Boole, and then the cube was placed below it in the boole (Figure 22). When active, the Boole subtracted the shape of the cube from the entire model, and the resulting model had a flat base (Figure 24).

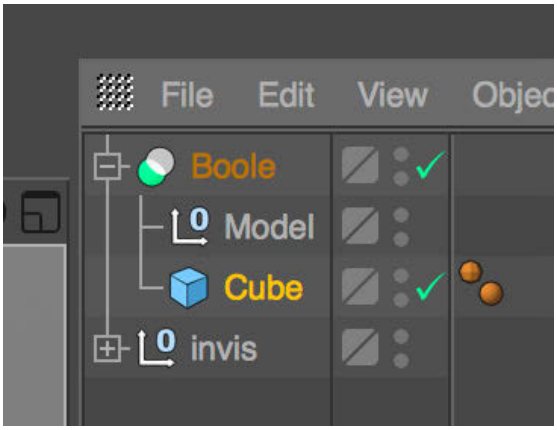


Figure 22. C4D boole hierarchy.

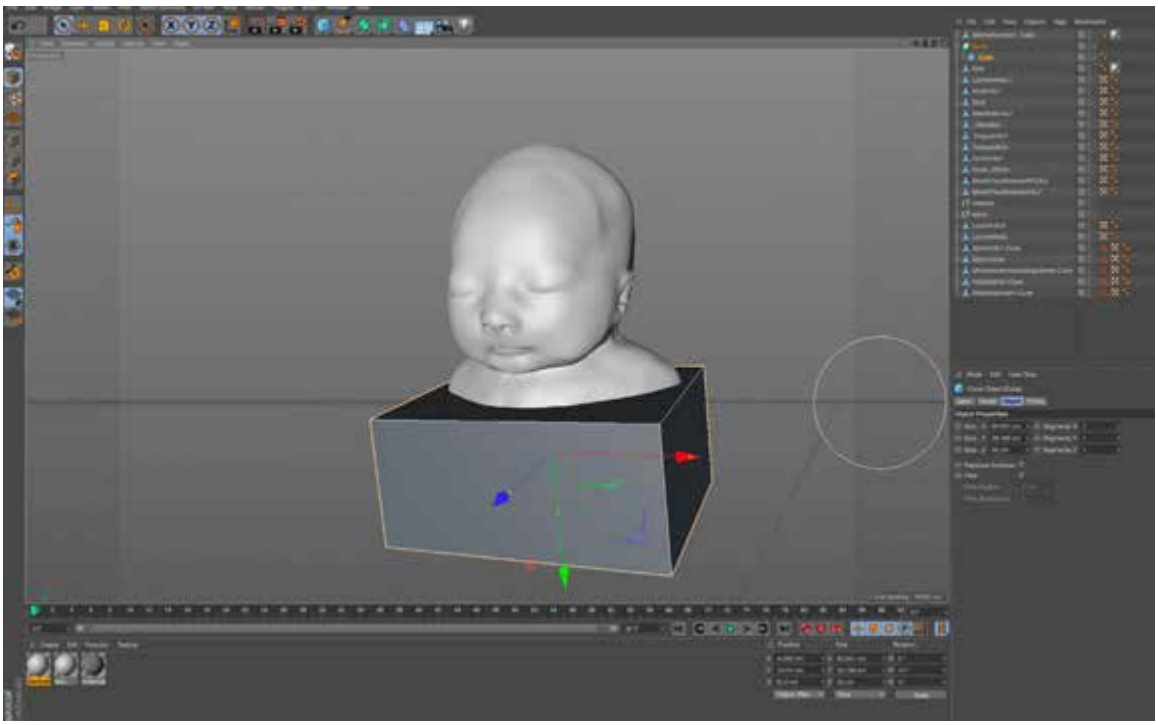


Figure 23. C4D boole model *(text not intended to be read).*

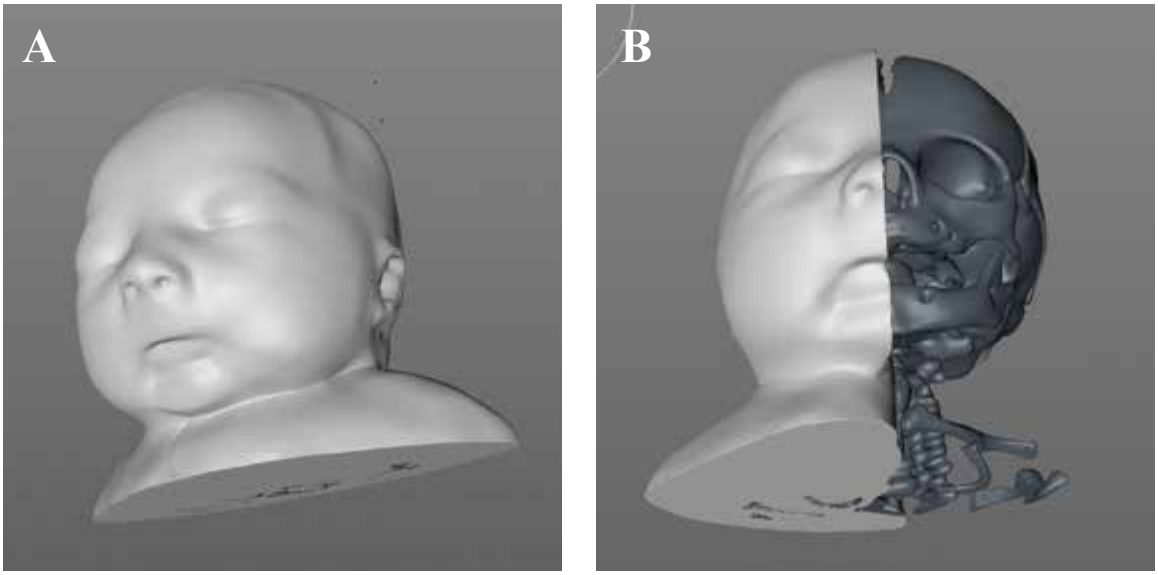


Figure 24. Model with flat base. **(A)** Entire skin. **(B)** Sagittal skin.

The final model was exported as an FBX file from C4D.

Interactive Application Design

Storyboards for each scene of the application was drawn in Procreate for iPad Pro at the beginning of this project. These storyboards were used as loose designs for the creation of the application in Unity.

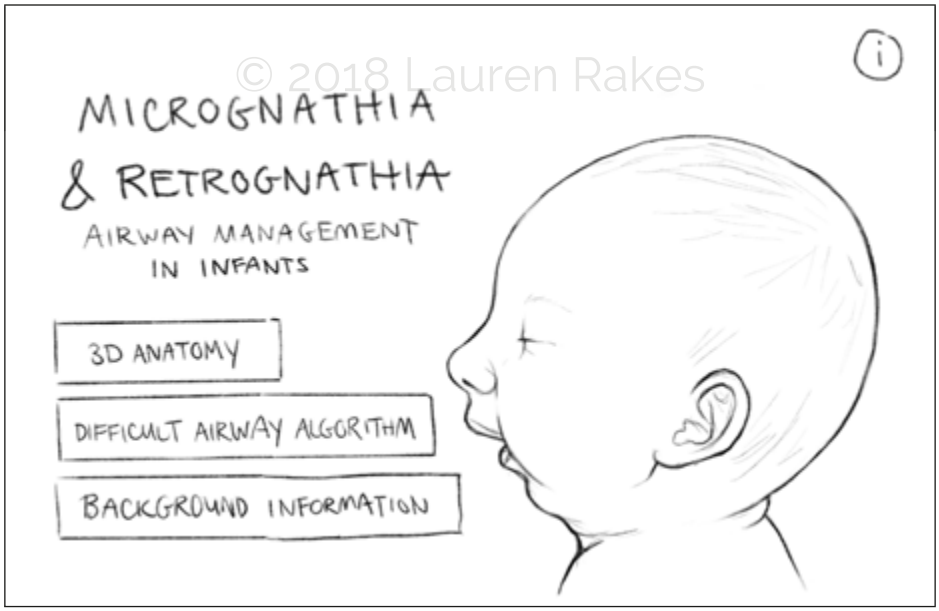


Figure 25. Sketched storyboard (Home Screen).

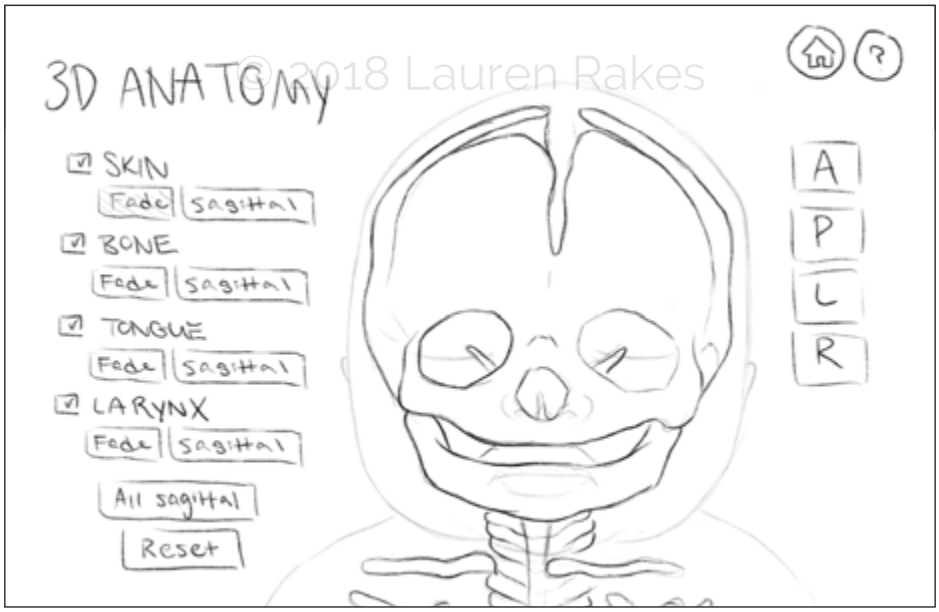


Figure 26. Sketched storyboard (3D Anatomy Section).

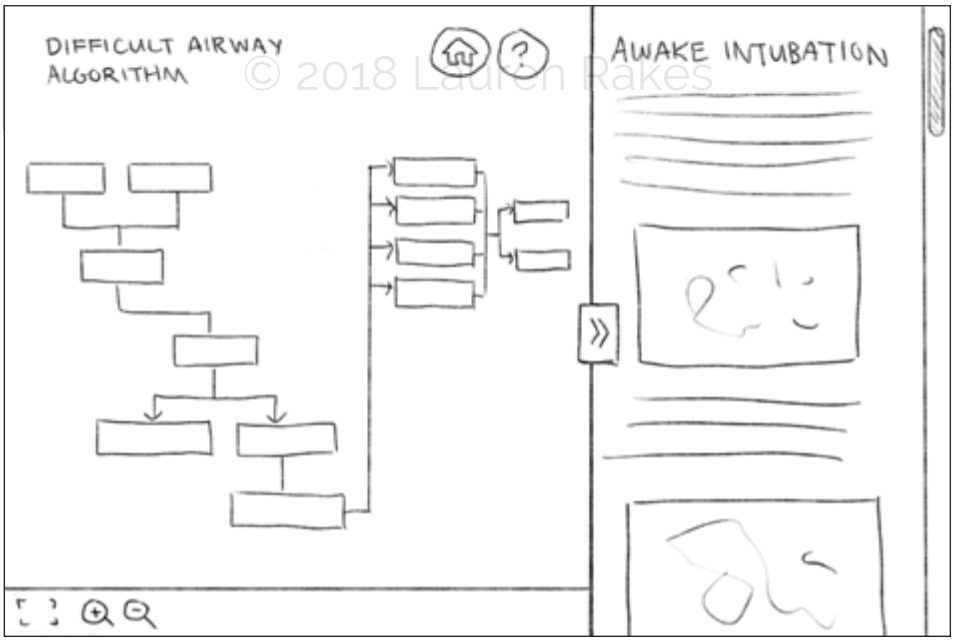


Figure 27. Sketched storyboard (Difficult Airway Algorithm Section).

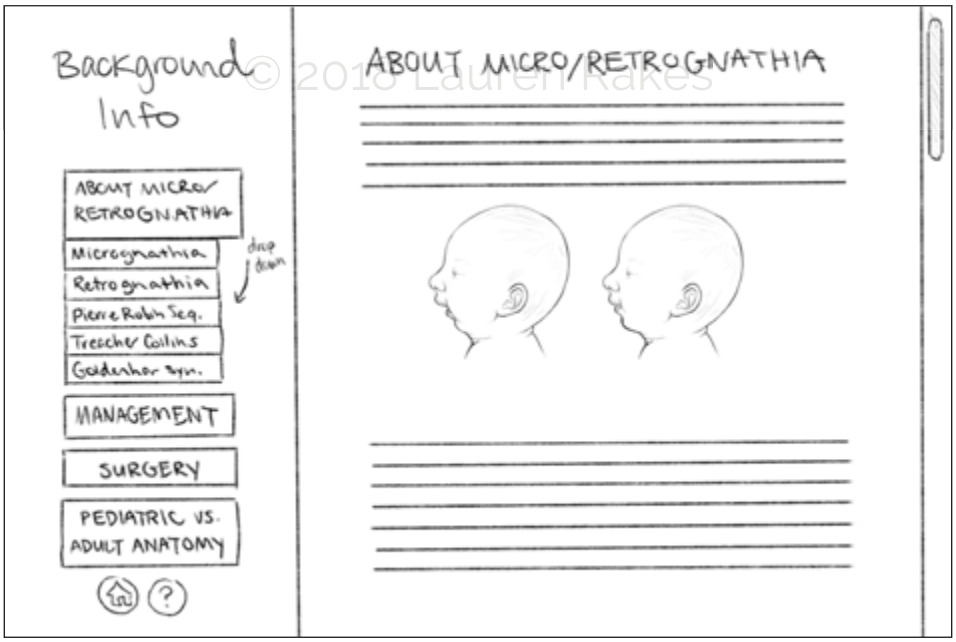


Figure 28. Sketched storyboard (Background Information Section).

Interactive Application Creation in Unity 3D

Unity (Unity 2017, Unity 3D) is a game development platform that supports building games or interactives that can be played on up to 27 different systems, including Mac, Windows, iOS, Android, and WebGL. Unity comes in Personal, Plus, and Pro versions. The Personal version is free, but has several built-in limitations, such as the inability to play movies or animations. However, the application for this project was created utilizing the free Personal version, since it still allowed the implementation of most of the features planned in the original design.

Unity was chosen over other game development platforms because of the easy accessibility of a wide range of user-friendly resources ("Unity User Manual" 2018). Additionally, the documentation for Unity is thorough and updated regularly. Unity is commonly used by both beginners and professionals to build a variety of 2D and 3D games or applications for almost any platform. It is possible to export a Unity project in WebGL format, allowing the application to be played from any web browser without the use of Flash media player. Unity allows for the integration of 2D and 3D scenes into one application, a feature necessary for this project.

Scripting in Unity

Unity uses a combination of Unity Events and scripting to tell the application how to run. Unity can function with C# and Javascript. C# was chosen for this project because of personal preference. Unity's own online Unity Manual provided information used to write the scripts for this project. Other scripting solutions were solved by referring to online tutorials on YouTube and Unity forums.

Unity Projects

Once Unity has been downloaded, a new project is created by clicking on **New** and **Create Project** (Figure 29). Upon opening the project, a new **scene** is created. Additional scenes can be created with File > New Scene.

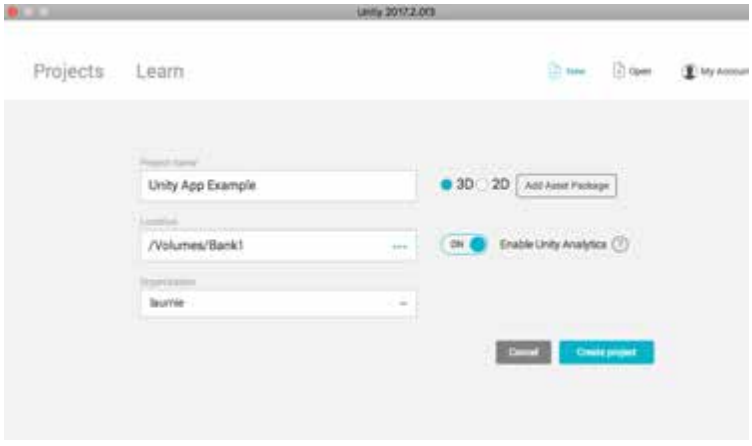


Figure 29. New Project window *(text not intended to be read).*

Unity collects assets in an assets folder in the **Project** sidebar. To keep assets well organized, folders were created for **Animations**, **Scenes**, **Scripts**, **Materials**, **Sprites** and **Art**, **3D Model**, and **Fonts**. All of the assets in the application were built in Unity except for the 3D model of the Pierre Robin sequence baby and the additional illustrations.

Importing 3D Model

To import the 3D model, a new **scene** was created to contain the 3D Anatomy section of the application. The FBX file of the model was imported (Assets > Import New Asset) and placed in the 3D Model folder (Figure 30). Clicking and dragging the model into the **Hierarchy** panel placed it into the 3D space of the scene. To easily manipulate the model and all of its pieces, it needed to be placed into an empty **GameObject** that had an axis of rotation centered on the model. To do this, a new GameObject was created (GameObject > Create Empty) and positioned in the center of the 3D model. When centered, the 3D model was placed as a child of the GameObject, which allowed all parts of the model to be manipulated using the GameObjects coordinates and rotation (Figure 31).

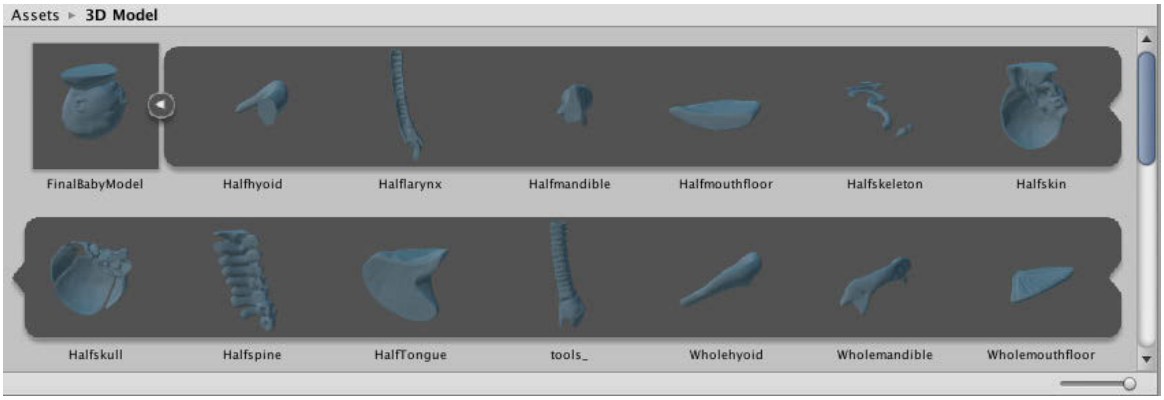


Figure 30. Asset window, 3D Model folder *(text not intended to be read).*

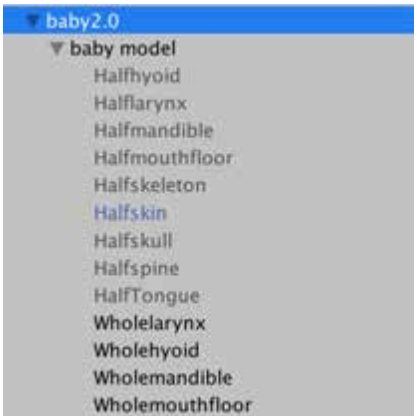


Figure 31. 3D model hierarchy *(text not intended to be read).*

Inspector

The **Inspector** houses all the information that needs to be accessed when a GameObject is selected in the hierarchy panel of the scene (Figure 32). From the Inspector, the position, rotation, and scale of a GameObject can be adjusted. **Components** can also be created, which add effects to the selected GameObject. The most common components created for this project were a **Box Collider** and all of the various **C# scripts**. At any point, new components can be added by clicking the **Add Component** button at the bottom of the inspector or by dragging a component from the Project panel to the Inspector.

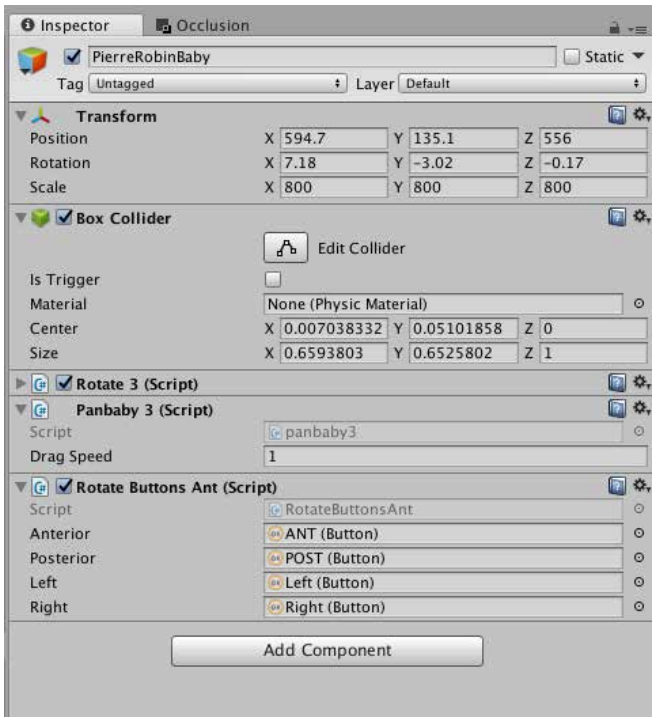


Figure 32. Inspector window *(text not intended to be read).*

Scenes

Each part of the application was broken down into scenes. The application created for this project had four scenes: **Main Menu, 3D Anatomy, Difficult Airway Algorithm,** and **Background Information**. By creating a script (Apendix A) that opens a new scene and attaching it to a button, a user is able to change between scenes during the runtime of the application. When building the application, (File > Build Settings) each scene must be added to the **Build Settings** in order for them to be included in the final application (Add Open Scenes). "WebGL" was selected in the Build Settings in order for the application to built in that format (Figure 33).

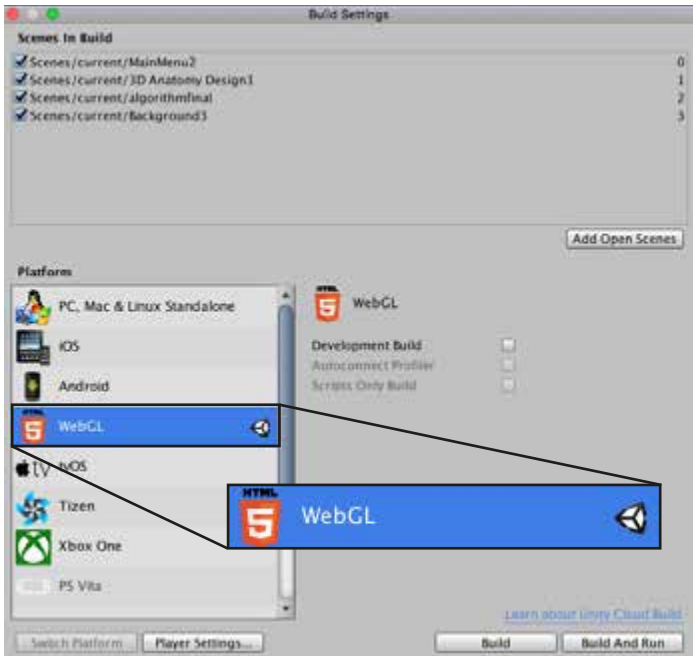


Figure 33. Build Settings *(text not intended to be read).*

User Interface

The **User Interface** (UI) system within Unity allows one to create panels, buttons and other elements that allow the user to interact with elements of the application. Some UI elements were created within Unity, and others were designed in Adobe Illustrator and brought into unity as **PNG** assets then converted to **sprite** (2D and UI) textures. These assets were often placed into a scene with a **Button** component attached allowing the user to click and interact with the sprites.

Canvas

All UI elements must be children of a **Canvas** element to appear in the scene. In the Canvas Inspector, the Canvas' **Render Mode** can be set to **World Space, Screen Overlay,** or **Camera Space** (Figure 34). World Space allows canvas objects to be placed into the scene among other assets. The camera can move around World Space elements in the scene. Camera Space and Screen Overlay both act similarly. They display the Canvas elements overtop of all of the world space elements in the

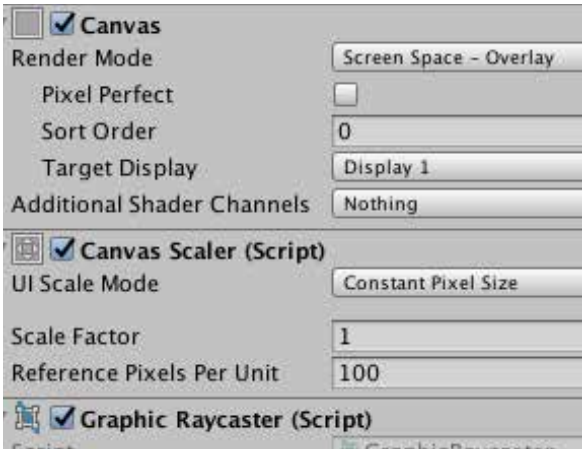


Figure 34. Canvas settings within inspector *(text not inteded to be read).*

scene. This is useful in placing buttons and information panels for the user interface.

Button

The Unity UI System includes customizable buttons. These are useful tools with a Button Script already attached that makes adjusting **Highlighted Color, Normal Color,** and **Pressed Color** possible inside the inspector (Figure 35). Buttons also have an additional **OnClick () Event** Component that allows some events and actions to be assigned without scripting. Customizable buttons were used in cases where a GameObject had to be turned on and off with a button click. By using OnClick () Events, the GameObject was dragged directly into the component in the Inspector and the **SetActive** action was assigned. Scripting and OnClick events were used to achieve various effects, such as toggling meshes, camera navigation, scene loading, material changing, and more.

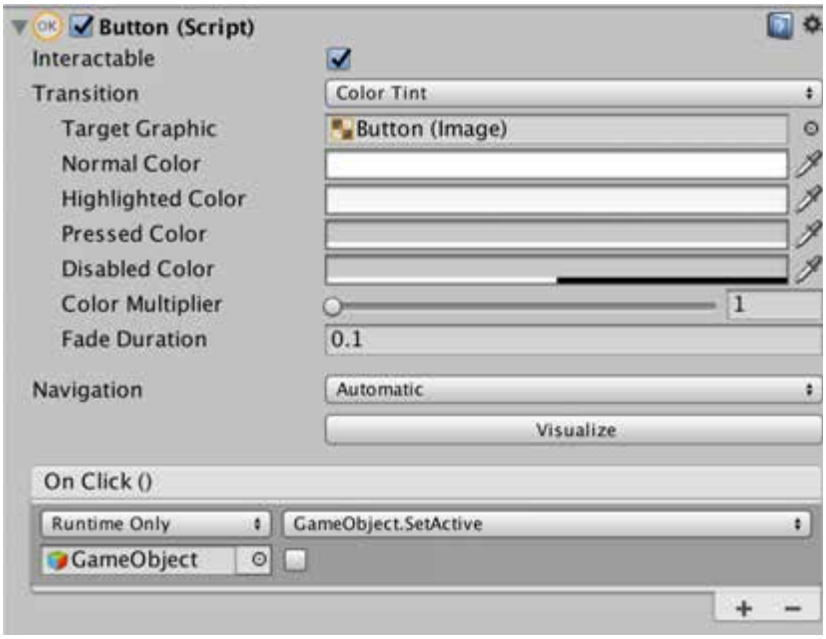


Figure 35. Button script and OnClick () Event Component *(text not intended to be read).*

Basics Code Syntax and Explanation

Each script starts with 3 lines of code:

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
```

These lines import **namespaces** that collect classes and other data to allow the rest of the script to work. Next, a **public class** is defined, and is always named the same name as your C# Script file.

```
public class Rotate3 : MonoBehaviour {}
```

Within these brackets, the "action" of the code is written. Various public classes and private classes can be defined depending on the script. **Classes** are treated as blueprints, and define "certain properties, fields, events, method etc" (TutorialsTeacher 2018). **Public** makes the variable visible in the inspector of the object on which the script is attached, and private makes it only visible in the script.

```
private float rotationSpeed = 5.0F;
```

Void start () {} indicates that whatever script goes into the bracket will run as soon as the script is called upon. This space can be used to define objects, attach materials, name EventListeners, etc.

Scripts within **void update ( ) {}** run every frame during runtime. Because of this, scripts that are taxing on the performance of the application should not be put within void update ( ). Instead **If** statements should be used. For example, If(Input.GetMouseButtonDown(1)){} will detect a right mouse click and execute the script within the brackets only when the right mouse is clicked.

Void OnMouseOver() and **void OnMouseExit()** allows script actions to execute when the mouse enters either a box collider space on a GameObject or enters the space of a UI Button.

Script examples can be found in the Appendix.

OnClick Events

The **OnClick ()** event component is an feature in Unity 3D that allows actions and scripts to be attached to buttons when they are clicked without writing a script and attaching it separately (Figure 36). Use of OnClick () events saved time and felt much more intuitive because of its drag-and-drop nature. The main use of OnClick () events was to turn certain GameObjects on and off while the application was running. This was done by dragging the GameObject into the OnClick space in the inspector, choosing the name of the GameObject under the drop down menu, then selecting **SetActive bool**. This created a boolean operation for the game object. A check box appeared next to the name of the object in the OnClick () section. If checked, the GameObject would become active and visible when the button that contains the

OnClick () component is clicked. Multiple GameObjects can be assigned in each OnClick () component.

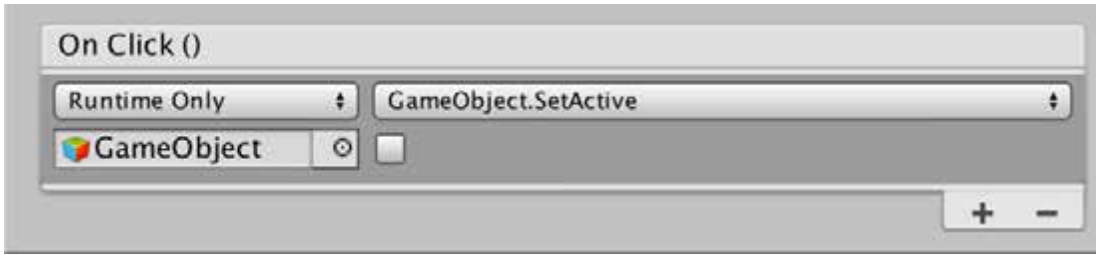


Figure 36. OnClick () Event.

Camera

The default **camera** was used for all scenes. Elements of the camera, such as **position, rotation, field of view,** and **type of view** (orthographic vs. perspective) varied depending on the scenes. In some cases, a script was attached to the camera to move it along its Z plane or change the field of view when the mouse wheel's value was greater or lesser than zero, thus creating a zoom action (Appendix B and Appendix C). Another script was applied to the camera to produce a method for panning around the scene along the X and Y plane but not the Z plane (Appendix D).

Lighting

Default lighting was used for each scene. Lighting only affected elements in the 3D Anatomy scene, not in the other scenes that only contained 2D elements. The light was placed in the upper left hand corner, which is standard for medical illustrations and animations. A key light was placed in the lower right to

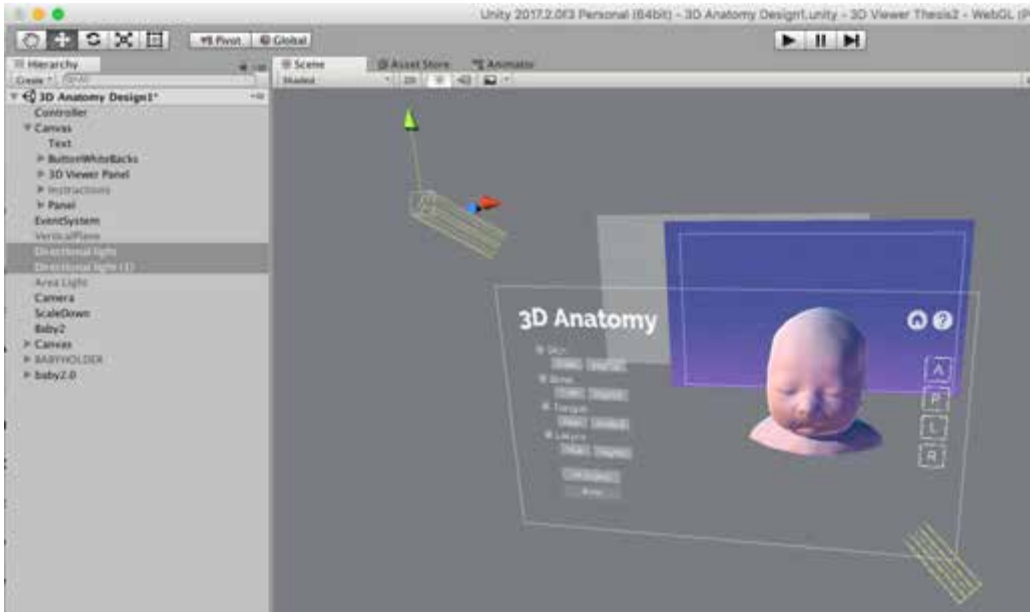


Figure 37. Lighting setup *(text not intended to be read).*

provide some reflective light (Figure 37).

3D Anatomy Materials

Parts of the model needed to turn transparent during application runtime for the user to fully understand the anatomy beneath. Two materials were made for each part of the 3D model: one "normal" material using an opaque shader set to 100% alpha, and one "transparent" material using a transparent shader set to 0% alpha (which appeared almost completely transparent, but didn't disappear). A script was created to toggle between the two materials upon a button click (Appendix G).

In the script, a public class was created for each material, named **Mat1** and **Mat2**. In the Inspector, the corresponding material was dragged into the slot for Mat1 and Mat2. Next, a **public bool** was created to house a **true** or **false** state for each material. The first material's variable for the bool was named **FirstMaterial** and was set to true. The second material's variable was named **SecondMaterial** and was set to false. A **public button** was named and assigned in the Inspector and to trigger the material change.

Under Void Start (), an event listener was created with the name **TaskOnClick** and assigned to the button. Under the event listener TaskOnClick, the bool was defined. If the objects had FirstMaterial attached, then upon click, SecondMaterial was attached. SecondMaterial was then assigned a true state, and FirstMaterial was assigned a false state. In the "**else if**" statement, the opposite was declared, so that if the second material was on the objects, the first material would be attached on button click.

A similar script was written for each type of material: Skin, bone, tongue, and larynx. An On Click () event was made to toggle on a blank shape behind each clicked button. If the blank shape was there, then the button had already been clicked and was "active" and appeared more opaque compared to an unclicked button. This signals to the user that the button has already been activated (Figure 38). Upon clicking the button again, the blank shape toggles off and the button returns to normal.

Figure 38. Active button state.

Visibility Toggle

All elements in each scene have a toggle for visibility (Figure 39). This was useful for some parts of the application where objects needed to be turned on and off. By attaching a script to a controller or button that sets the object's active state to true or false, the visibility of the object can be toggled during runtime.

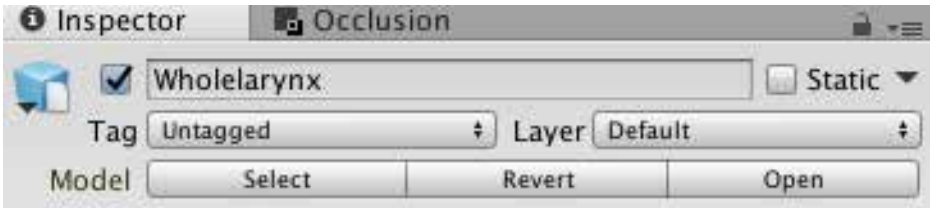


Figure 39. Visibility toggle in Inspector.

3D Anatomy Navigation

Within the 3D Anatomy section of the application, the user can zoom into the 3D model, pan, and rotate it. Three different scripts were written to handle these actions. To zoom into the 3D model, a script was written to move the camera's Z axis depending on the value of the mouse's scroll wheel (Appendix B). To pan the 3D model, a script was written to move the GameObject that holds the 3D model depending on position of the mouse when dragging (Appendix F). To rotate around the model, another script was attached to the GameObject holding the 3D model (Appendix E). This script takes into account the vector of mouse movement upon dragging to rotate the model, and retains a small amount of inertia upon mouse click release. This creates a more natural motion when the user stops rotating the model. In order for the mouse to interact with the 3D model, a **box collider** was created around the model (Figure 40).

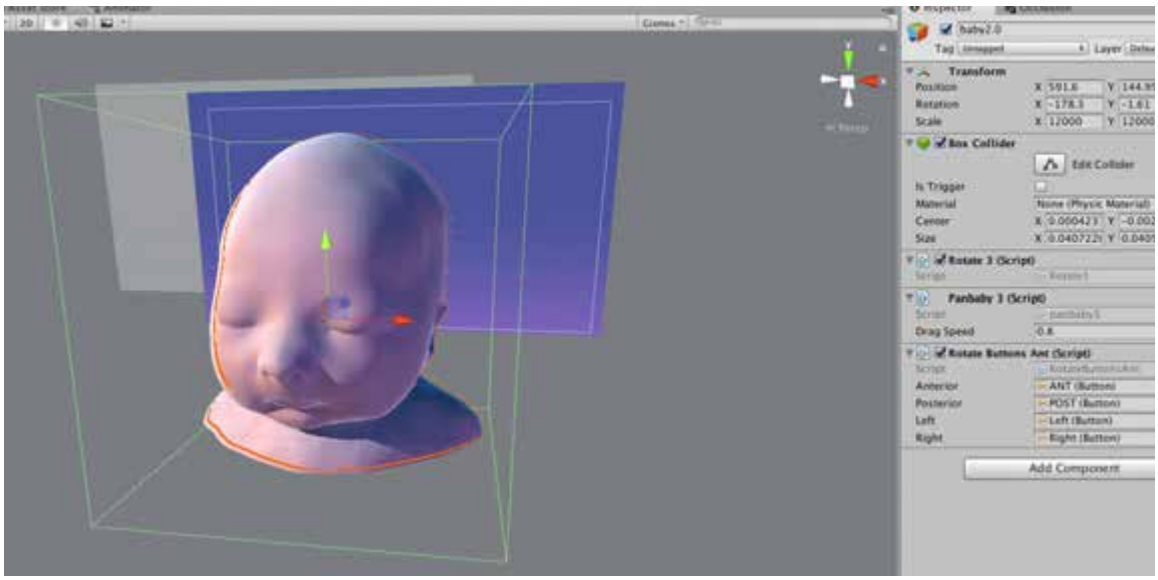


Figure 40. Box collider *(text not intended to be read).*

On the right side of the 3D Anatomy section, buttons were created that would align the model to predefined anatomical positions when clicked. The script defines the specific x, y, and z axis rotation values of the model in each position, and applies these values to the GameObject that contains the 3D model upon click (Appendix I).

To reset the view of the 3D model, an OnClick () event was written and attached to a reset button that simply reloads the scene, placing the model back to its original position (Appendix A).

3D Anatomy - Sagittal

In order to view the sagittal sections of each part of the 3D model, a script with a boolean operation was attached to each Sagittal button in the UI (Appendix H) (Figure 41). Upon start in the script, the "full" versions of the model were set to **true** and the sagittal versions of the model were set to **false**. When the sagittal button is clicked, the script checks whether the full version or sagittal version is active. If the full version is active, then it inactivates it and sets the sagittal version to true. If the sagittal version is active, then it inactivates it and sets the full versions to true. This way, the button can function as a type of toggle that switches between the sagittal and full versions of each model. The "All Sagittal" button in the UI simply turns all full elements to false and all half elements to true.

Figure 41. Script to change to sagittal skin model within the Inspector.

Difficult Airway Algorithm Navigation

Interactivity, user engagement, and aesthetics were considered in scripting the navigation of the Difficult Airway Algorithm section of the application. The algorithm consists of interactive buttons for each step, and sprites for the lines and arrows between the buttons. The whole algorithm is placed on a Canvas set to World Space, while the information panel and other elements sit above it on a Canvas set to Overlay. By clicking on the buttons in the algorithm, the information on the right panel updates and changes to the button's topic. Hovering over a button (or the corresponding button within the information

panel) turns on a yellow outline around the button, highlighting what is selected. The path leading up to the button turns yellow as well, and so do the outlines around the buttons in the preceding path (Figure 42). The information panel on the right always displays text and images relevant to the current selected button in the algorithm. Also in the information panel are buttons that allow one to choose the next step in the algorithm without having to go back and click on the actual algorithm (Figure 43). This allows the user to walk through the algorithm entirely within the information panel while only choosing the next available options.

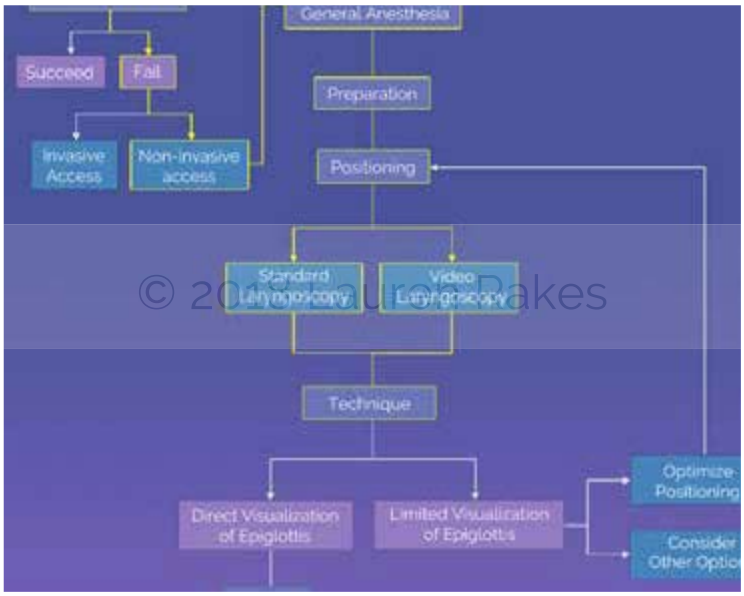


Figure 42. Yellow path in final algorithm *(text not inteded to be read).*

Figure 43. Buttons in the information panel *(text not intended to be read).*

The user can zoom into and pan around the flowchart to explore as if it were a piece of paper on a table. A script was written to move the camera's Z position depending on the value of the mouse's scroll wheel to zoom (Appendix C). Another script was created to pan the camera on its x and y plane, taking into account the mouse's position when dragging (Appendix D). A "Fit Screen" button was created to reset the algorithm to its original position without changing the information on the panel on the right. (Appendix J).

If the user toggles "Zoom Mode," the camera will zoom in a little closer and center itself each time a step in the algorithm is clicked. This action uses a similar script to the "Fit Screen" script, with camera positions defined for each algorithm button. If the application is being used on a small screen, Zoom Mode can be useful to view the small text.

To ensure that interacting with the information panel on the right does not affect the algorithm underneath, a box collider was created and placed over the information panel. A script was created so that when the mouse enters the box collider, the script to pan and zoom the camera is disabled. When the mouse exits the box collider, the script is enabled again (Appendix L).

Information Panels

The information panel on the right of the Difficult Airway Algorithm section of the application changes to correspond to each button/topic in the difficult airway algorithm. Each topic is contained in its own separate **Scroll View** component, placed in a single white background **Scroll Holder**. To change to each new panel's Scroll View, OnClick () Events were used (Figure 44). Each panel can either be set to "true" (visible during runtime) or "false" (not visible during runtime). There were 27 panels in total. Upon starting the Difficult Airway Algorithm section, only the information panel for "Awake Intubation" is set to active. The panels change either from a button click on the algorithm, or by clicking a button in the current visible Scroll View (Appendix K). Upon button click, all panels except the selected one are set to inactive, and the selected one is set to active.

Figure 44. On Click () Event that changes information panels.

Open online PDF

A PDF was compiled containing most of the text, images, and animations created for this project. In Unity, a script was written to open a new tab in the browser window currently running the application. The online PDF opens in the new tab (Figure 45). The new tab script was attached to buttons placed next

to corresponding headers in both the Difficult Airway Algorithm and Background Information sections of the application. By placing targets on certain sections of the PDF in adobe acrobat, hyperlinks can be generated that link to the targeted page on the online PDF. By assigning these hyperlinks to the buttons in Unity, the online PDF will open in a new tab to the page dealing with the subject the user wishes to explore.

The online PDF also has “Play Video” buttons next to some illustrations. Clicking these will open a relevant animation in a new tab for the user to view.

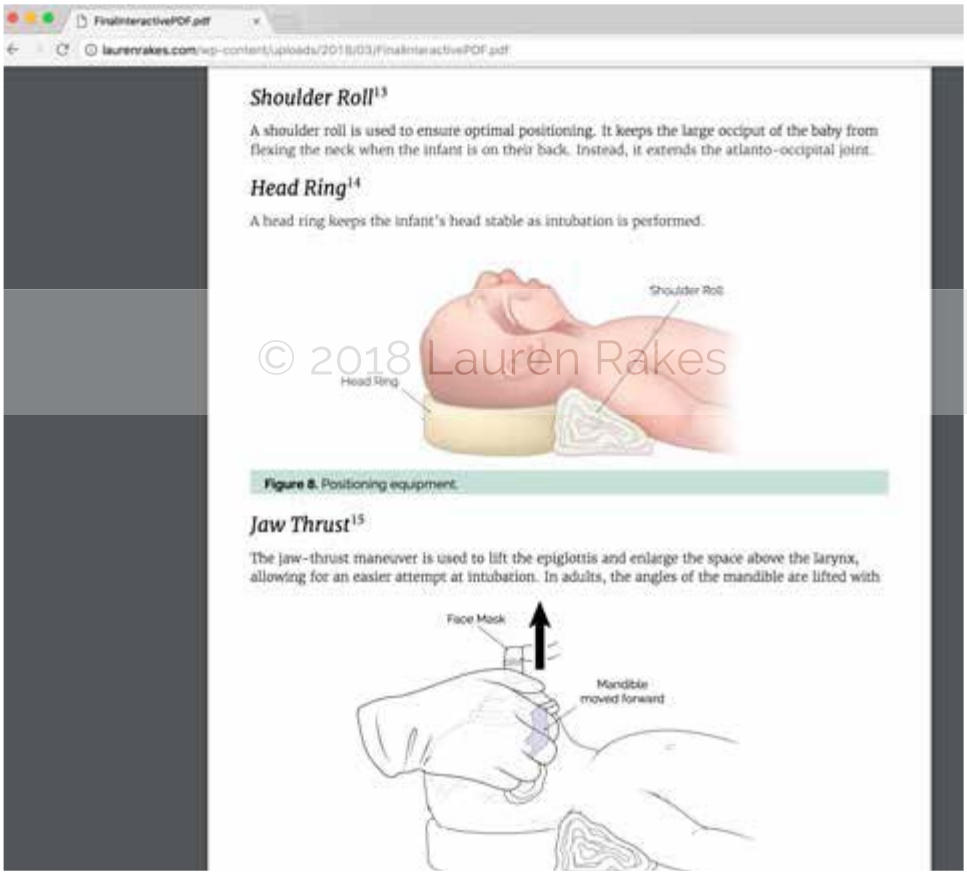


Figure 45. PDF within browser *(text not intended to be read).*

Flowchart Design

The difficult airway algorithm for micrognathic and retrognathic patients was developed by consulting with preceptors Dr. Deborah Schwengel and Dr. Jonathan Walsh. The unique algorithm was developed as a modified version of the adult Difficult Airway Algorithm already developed by The American Society of Anesthesiologists. The original algorithm was evaluated and elements were removed and added to fit the specific needs of micrognathic infants.

The flowchart was initially sketched on paper (Figure 46), and then mapped out using the flowchart software Draw.io (Figure 47). Revisions were made after consulting with preceptors, and then the flowchart was visually refined for use in the application with Adobe Illustrator (Figure 48).

In order for each part of the algorithm to be interactive, it had to be recreated with buttons and line sprites in Unity. The design created in Adobe Illustrator therefore was used as a reference to rebuild the algorithm in Unity.

Figure 46. Notes during flowchart development meeting *(text not intended to be read).*

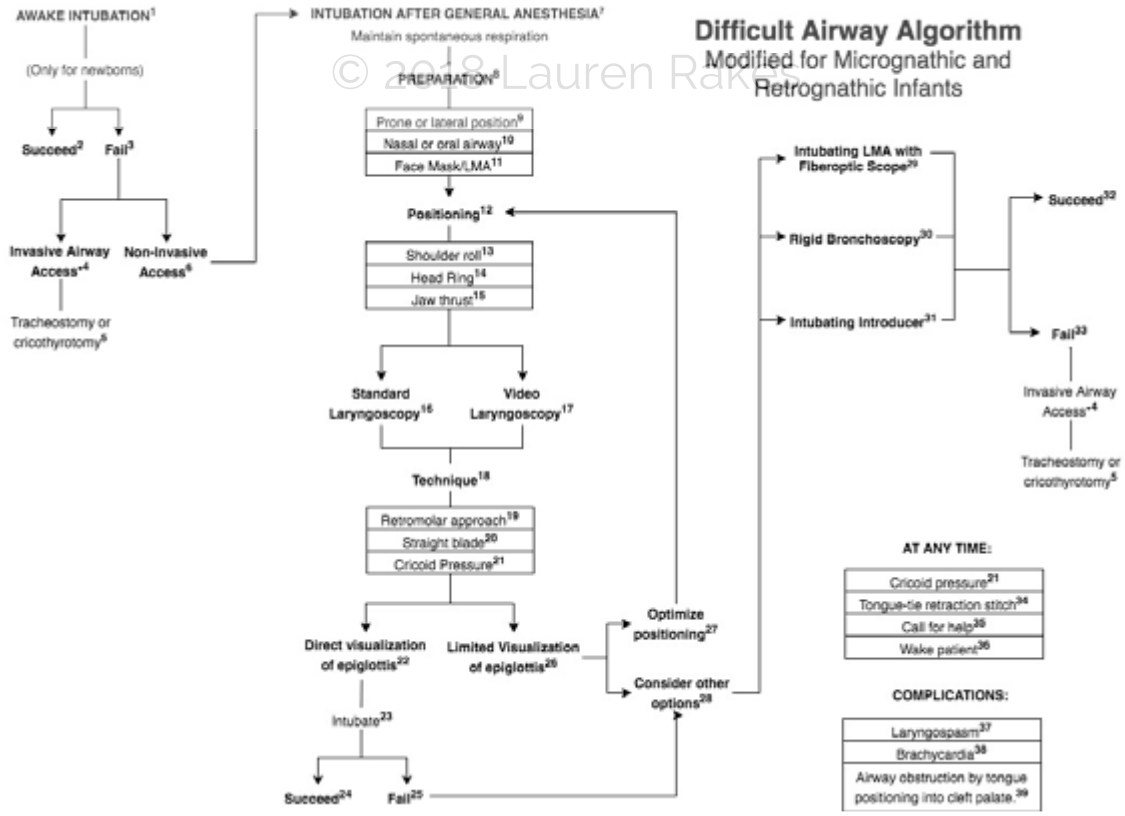


Figure 47. Draft of Difficult Airway Algorithm in Draw.io *(text not intended to be read).*

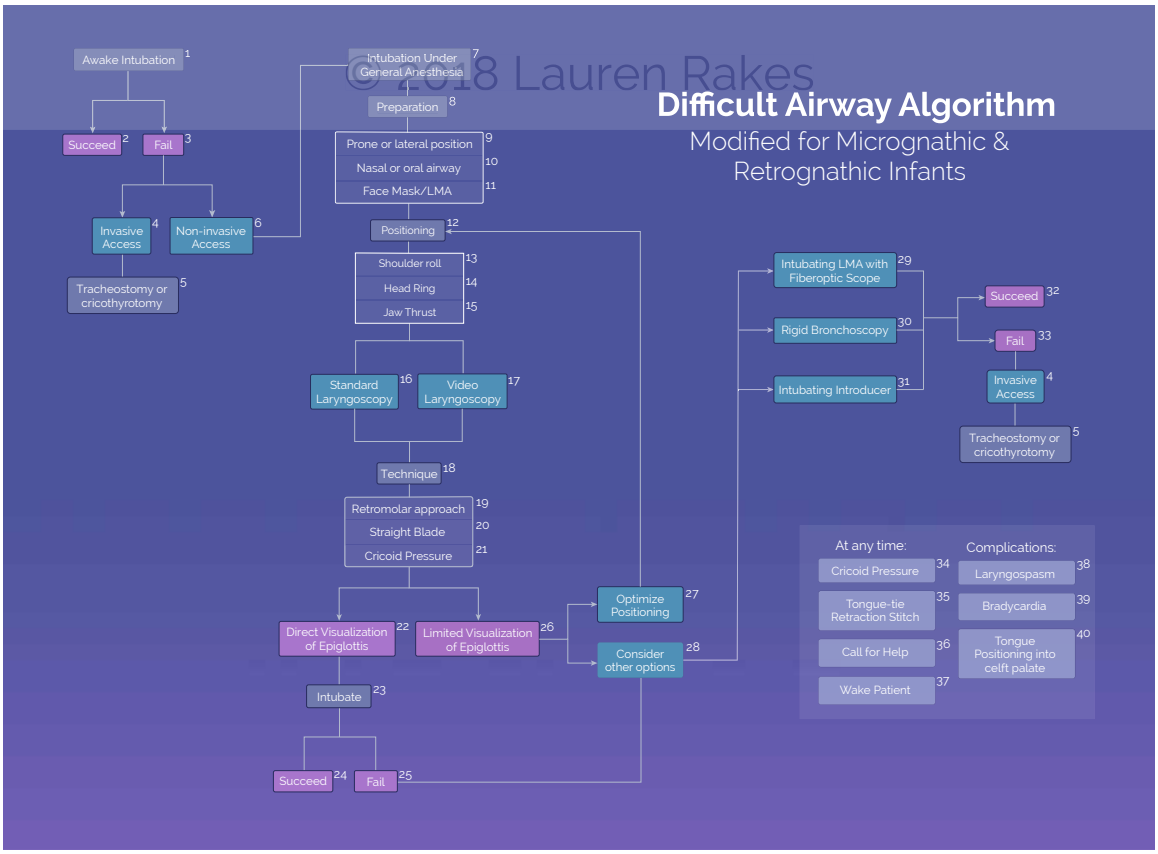


Figure 48. Difficult Airway Algorithm in Adobe Illustrator *(legible text in subsequent images).*

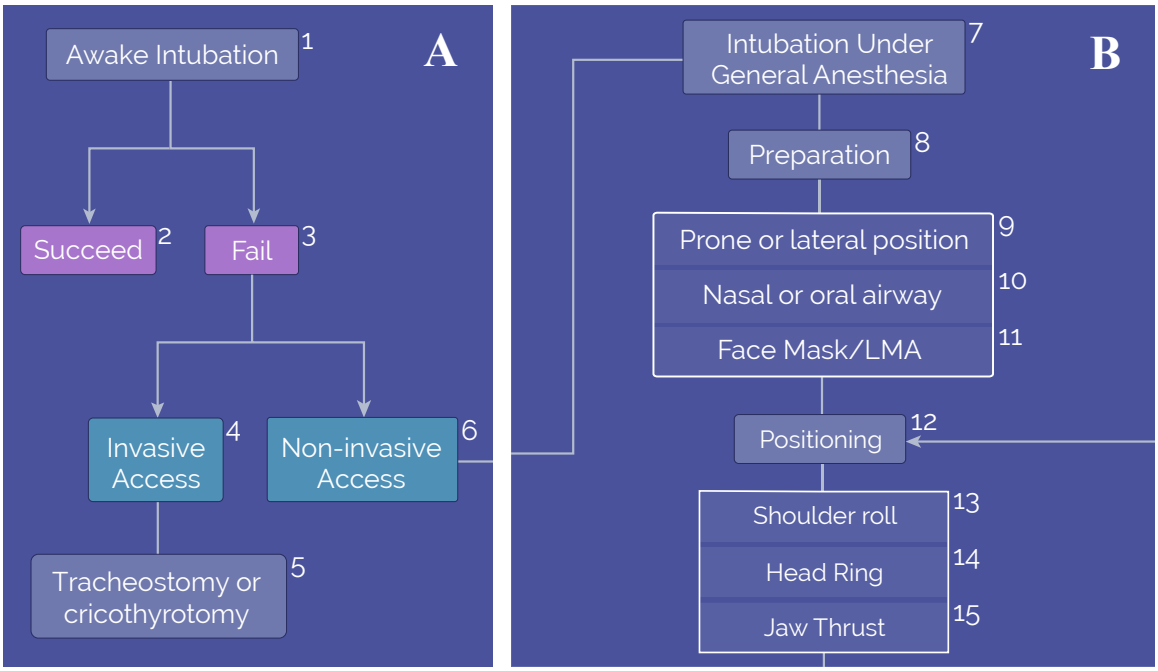


Figure 49. Details of Difficult Airway Algorithm in Adobe Illustrator, Part 1. **(A)** Awake intubation, etc. **(B)** Intubation under general anesthesia, etc.

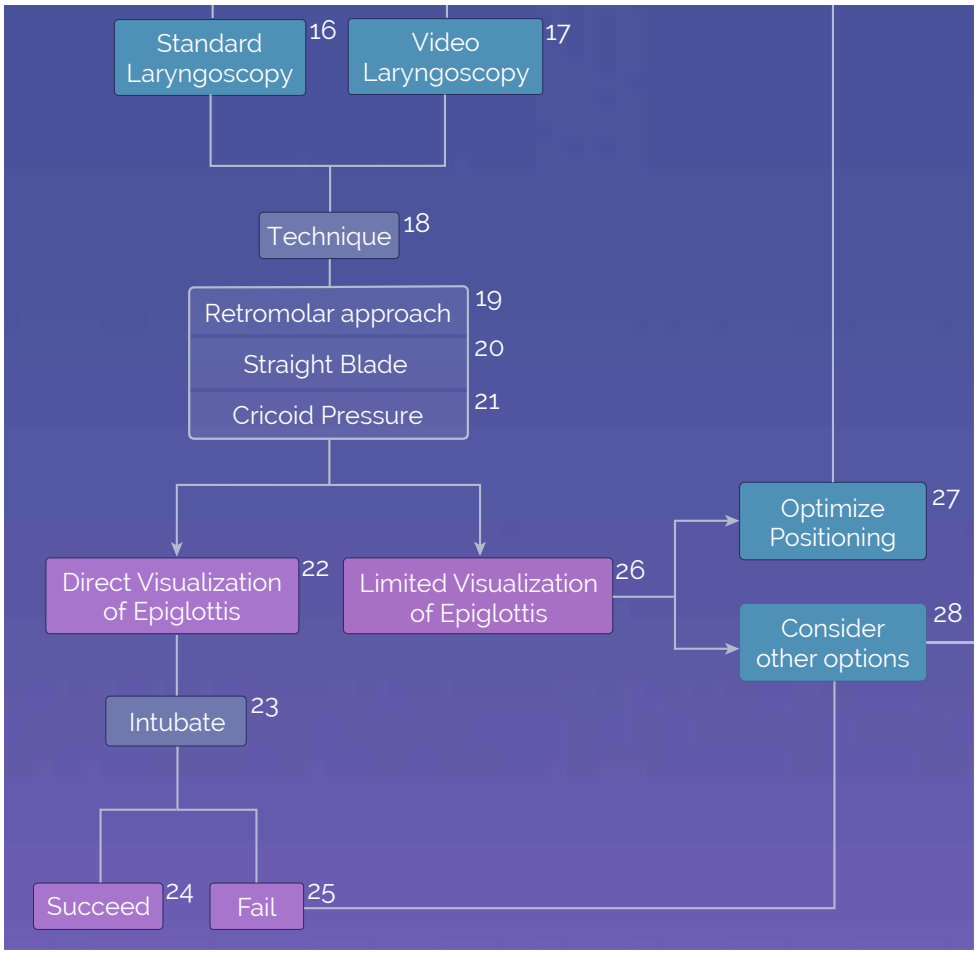


Figure 50. Detail of Difficult Airway Algorithm in Adobe Illustrator, Part 2.

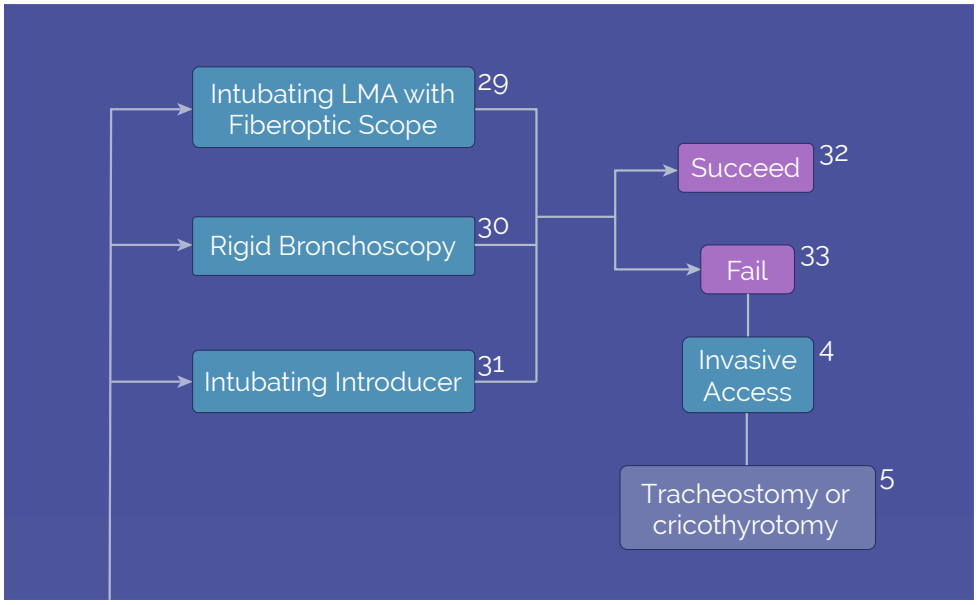


Figure 51. Detail of Difficult Airway Algorithm in Adobe Illustrator, Part 3.

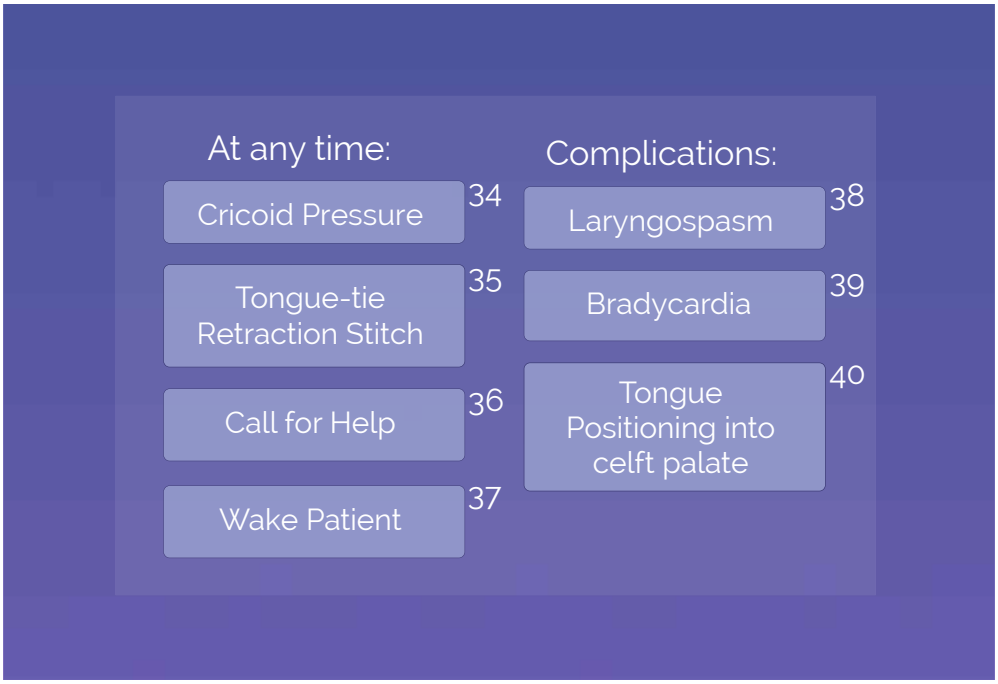


Figure 52. Detail of Difficult Airway Algorithm in Adobe Illustrator, Part 4.

Linework in Adobe Illustrator

Most of the illustrations generated for the application were first created as linework in Adobe Illustrator (AI). Some of the illustrations began as physical sketches scanned and imported into AI, others were created entirely in AI. In both cases, the pencil tool was used to either trace sketches or create the line and various stroke effects were then applied to taper lines and create the desired look (Figure 53).

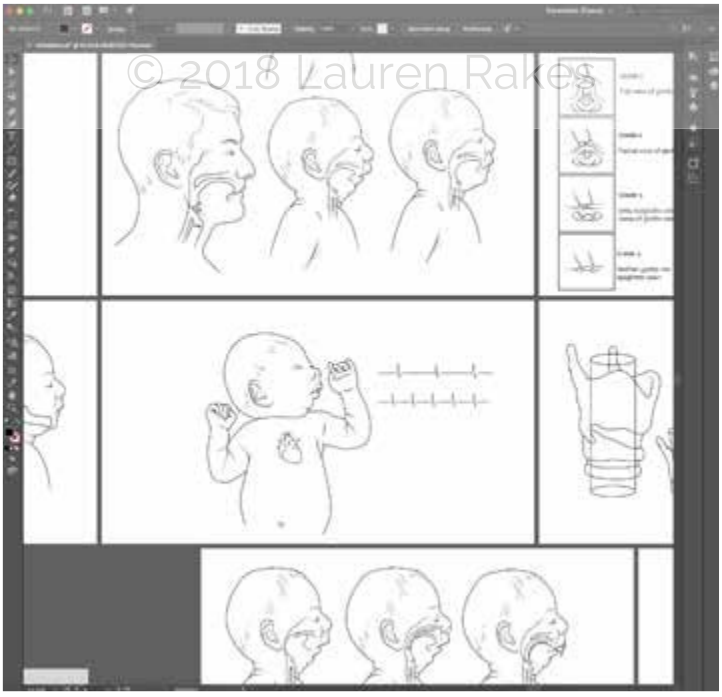


Figure 53. Linework in Adobe Illustrator *(text not intended to be read).*

Artwork Rendering in Adobe Photoshop

After the linework was completed for each illustration, it was imported into Adobe Photoshop (PS) for the addition of color and rendering. The linework was colored by clicking the **Lock Transparent Pixels** button with the lineart layer selected. Simple rendering was done with a soft airbrush (Figure 54).

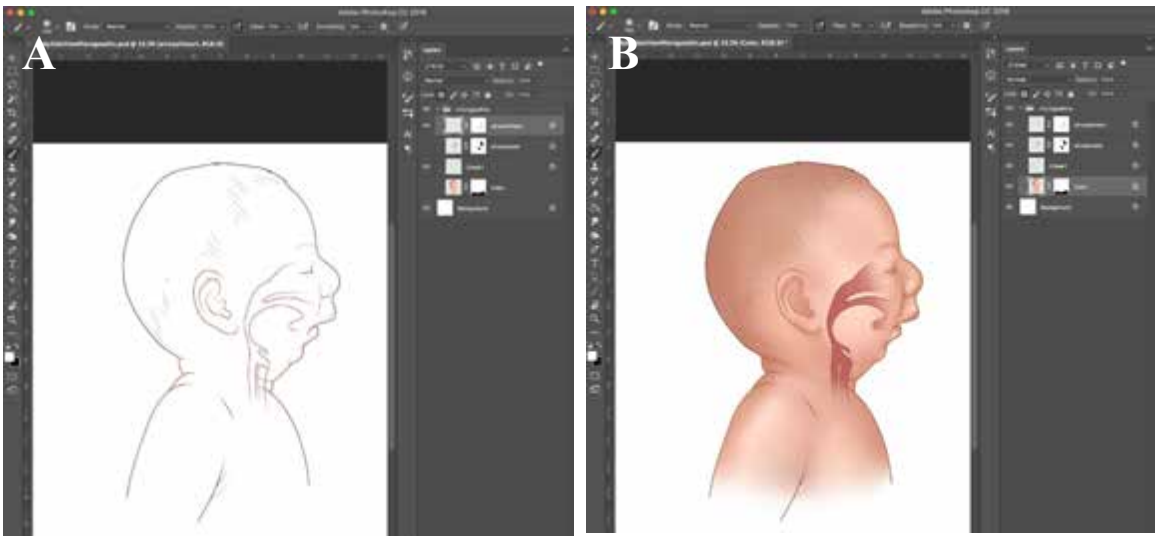


Figure 54. Rendering in Adobe Photoshop. **(A)** Lineart. **(B)** Color under lineart. *(Text not intended to be read.)*

Animation in Adobe After Effects

After the artwork was rendered in PS, it was imported into Adobe After Effects (AE). Animations were made in AE, primarily using the puppet warp tool and keyframing the position and opacity of layers

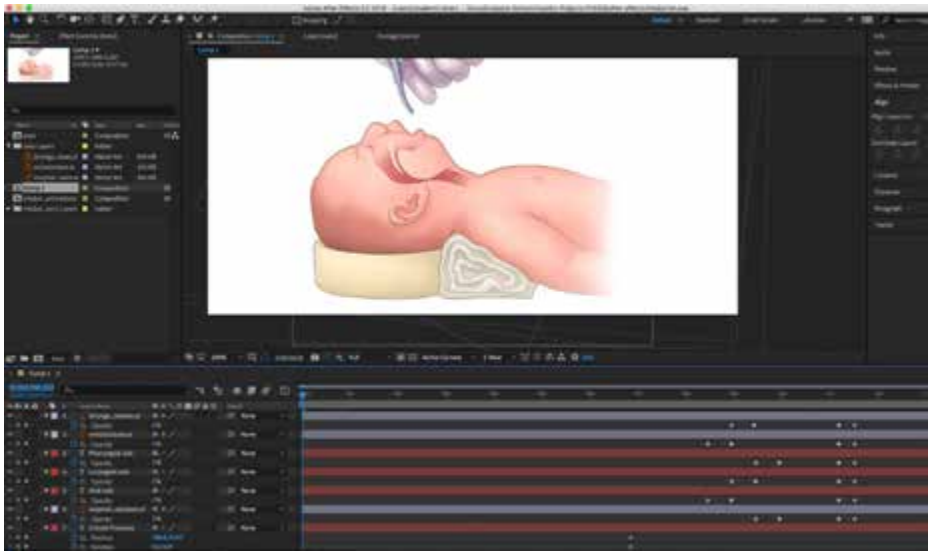


Figure 55. Animating in Adobe After Effects *(text not intended to be read).*

(Figure 55). After the animations were complete, they were exported to .mov format and uploaded to a website server. Buttons in the interactive online PDF were linked to the animations previously uploaded on the website, where they would open in a new tab for viewing.

Labeling in Adobe Illustrator

The artwork rendered in PS was brought back into illustrator for labeling. The typeface Raleway was used for all labels, as well as for the text within the application and online PDF.

PDF

Open online PDF

A PDF was created in Adobe Indesign (AIn), and contains most of the text and images/animations created for this project. A version of the PDF was uploaded to a web server so that it can be accessed through the application. Targets were placed on certain sections of the PDF in Adobe Acrobat (AA), which generate hyperlinks to the targeted page on the online PDF. By assigning these hyperlinks to "new tab" buttons in Unity (Figure 56), the online PDF will open in the new tab and take the user directly to the desired page.

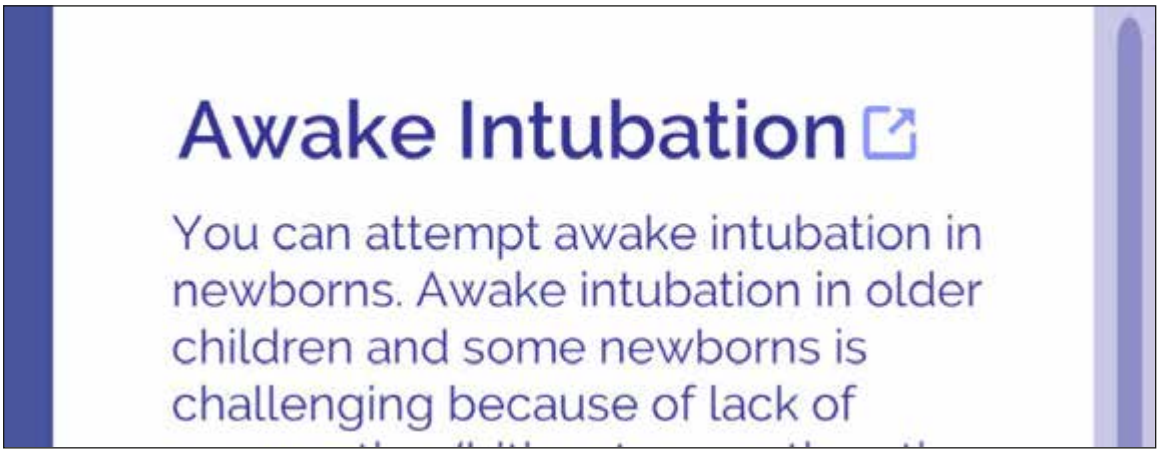


Figure 56. Open PDF button.

The online PDF also has several "Play Video" buttons next to select figures (Figure 57). Clicking these buttons open a relevant animation in a new tab for the user to view.

Figure 57. Play Video button.

Four full-color booklets containing the contents of the PDF were printed using www.blurb.com to supply to physicians in the Department of Pediatric Anesthesiology at Johns Hopkins Hospital.

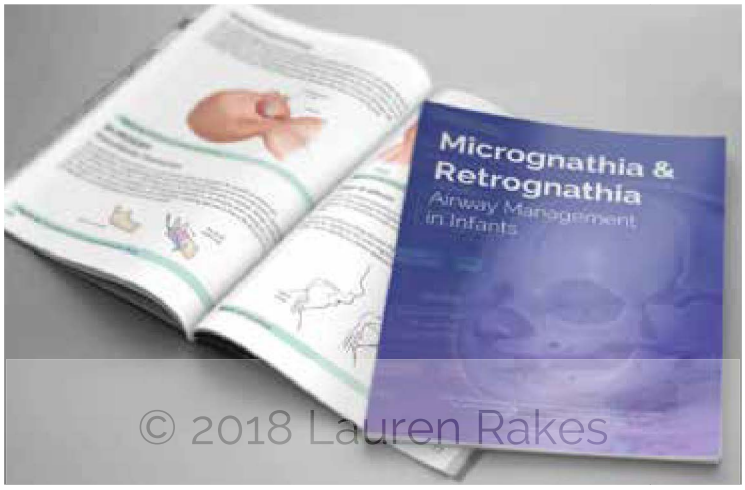


Figure 58. PDF booklet *(text not intended to be read).*

3D Rendered Stills

Materials and lighting were created in Cinema 4D (C4D) and attached to the Pierre Robin sequence 3D model. The materials made for the skin and the tongue were based on pre-made materials available in C4D ("Skin 01" and "Gums" respectively). Subsurface scattering within the luminance texture channel in the material settings provided a realistic and slightly translucent effect on the skin material (Figure 59).

Figure 59. Skin and tongue materials in C4D. **(A)** Skin material, luminance channel. **(B)** Gums material, color channel. *(Text not intended to be read.)*

The rest of the materials were created by starting with a base color and adding various levels of reflectance (Figure 60). One main light was created in the upper left, casting an area shadow (Figure 61). Three key light were placed strategically around the model's lower right side to illuminate the shadows and show any detail that would be lost otherwise. A camera was created with a focal length of 80 mm, the standard for portraits (Figure 62). TIFFs of the 3D model were rendered out from various angles (Figure 63). These images were used in the project's online PDF to provide another resource for studying micrognathic anatomy.

Figure 60. Skeleton and larynx materials in C4D. **(A)** Skeleton material, color channel. **(B)** Larynx material, reflectance channel. *(Text not intended to be read.)*

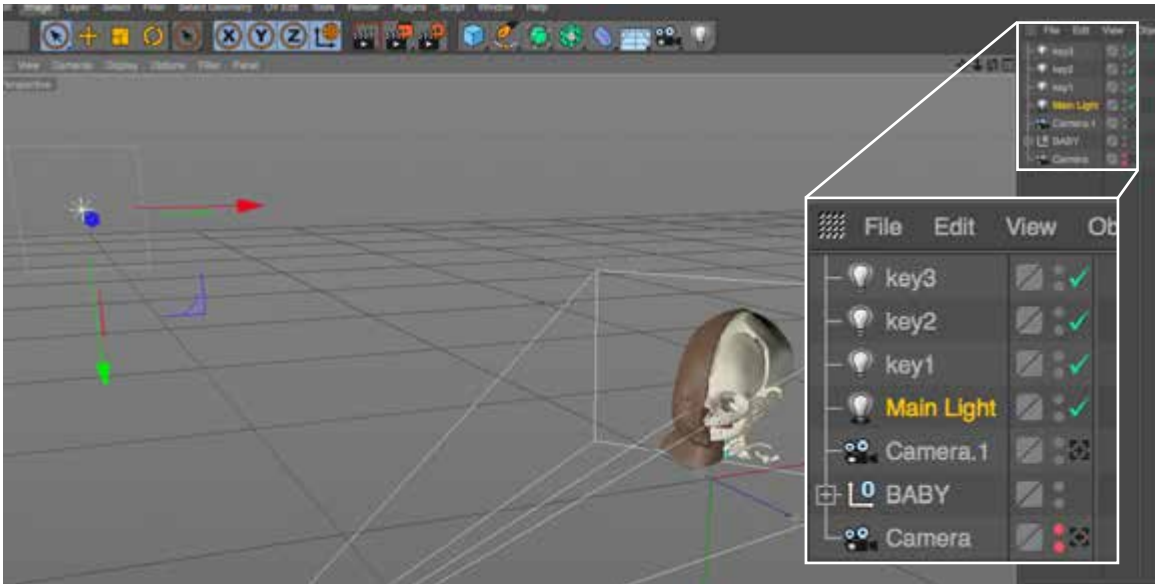


Figure 61. Lighting setup in C4D *(text not intended to be read).*

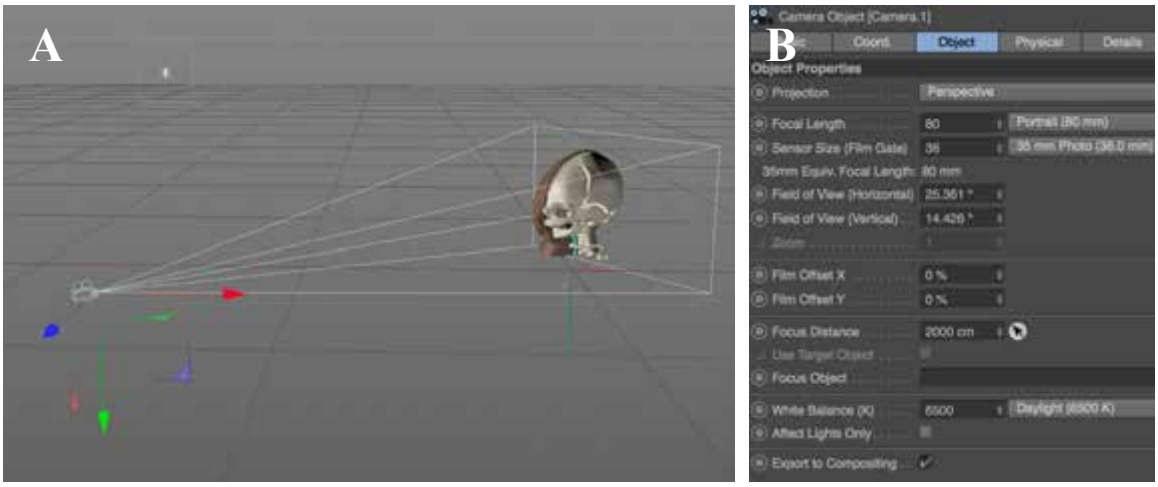


Figure 62. Camera setup in C4D. **(A)** Camera in perspective view. **(B)** Camera settings. *(Text not intended to be read.)*

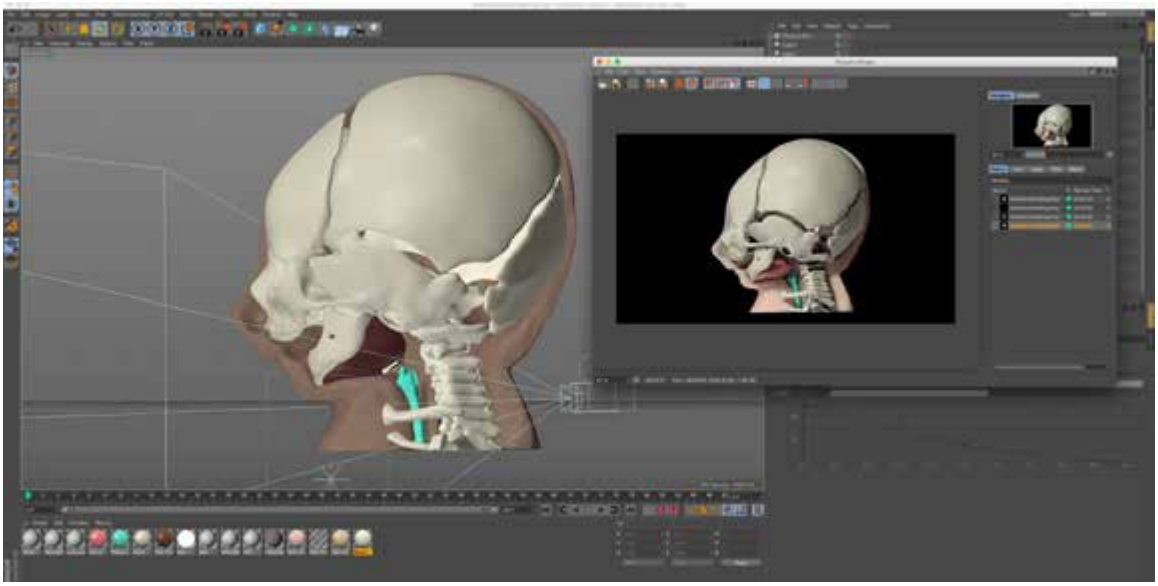


Figure 63. Rendering in C4D *(text not intended to be read).*

Results

3D Model

A 3D model was created using CT data of an infant with Pierre Robin syndrome and additional sculpting with Pixologic Zbrush and Cinema 4D. The 3D model consists of separate objects for the skin, skull, mandible, spine, skeleton (clavicle, parts of scapula, etc.), hyoid, tongue, mouth floor, and larynx . The skin and skeleton objects were based on surface renders taken from Osirix and the other objects were sculpted in Pixologic Zbrush. Each object has a complete version and a sagittal version. The sagittal skin object features a sculpted airway to show the airway obstruction typical in micrognathia.

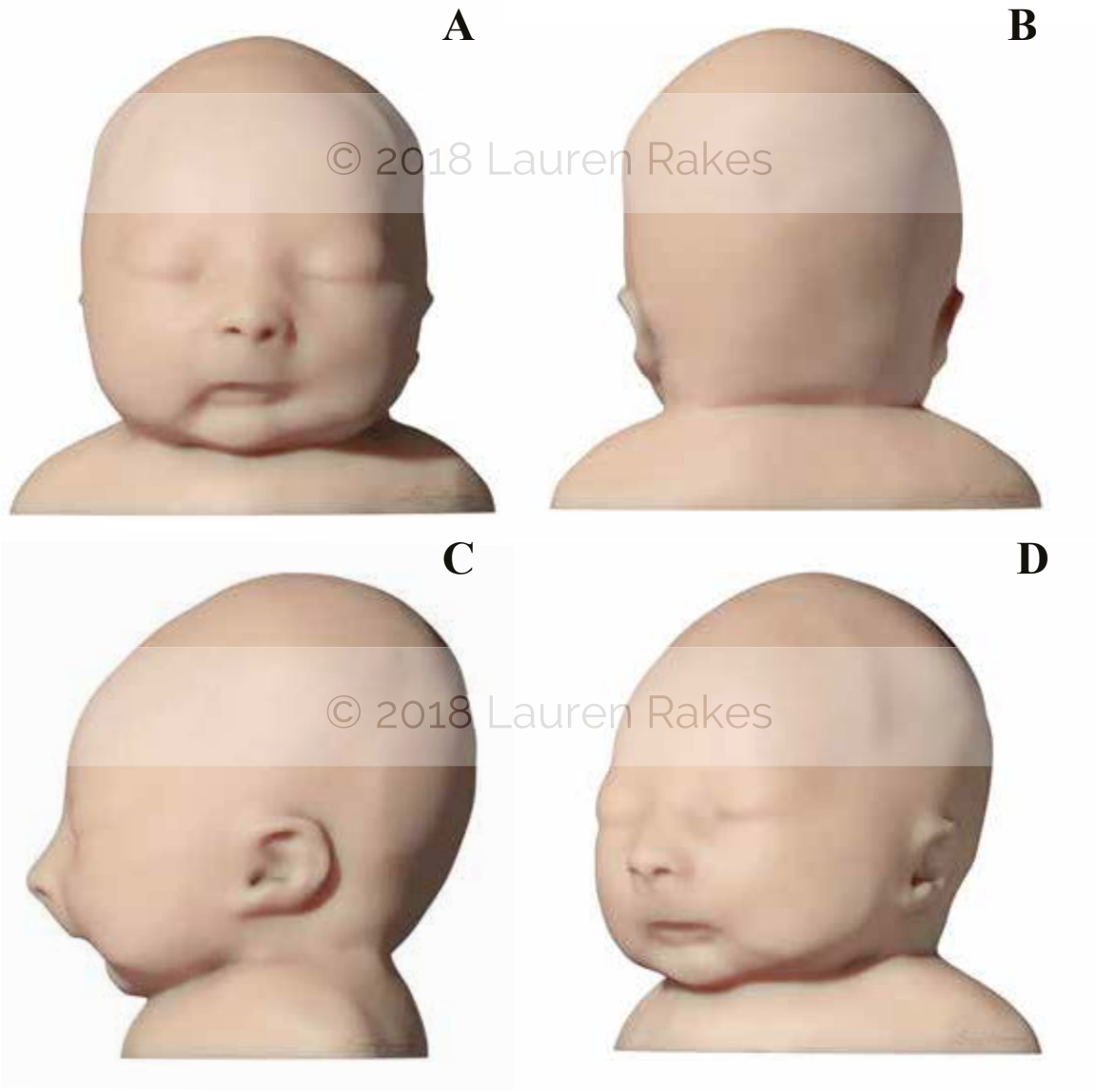


Figure 64. 3D model of Pierre Robin sequence infant. **(A)** Front. **(B)** Back. **(C)** Side. **(D)** 3/4 view.

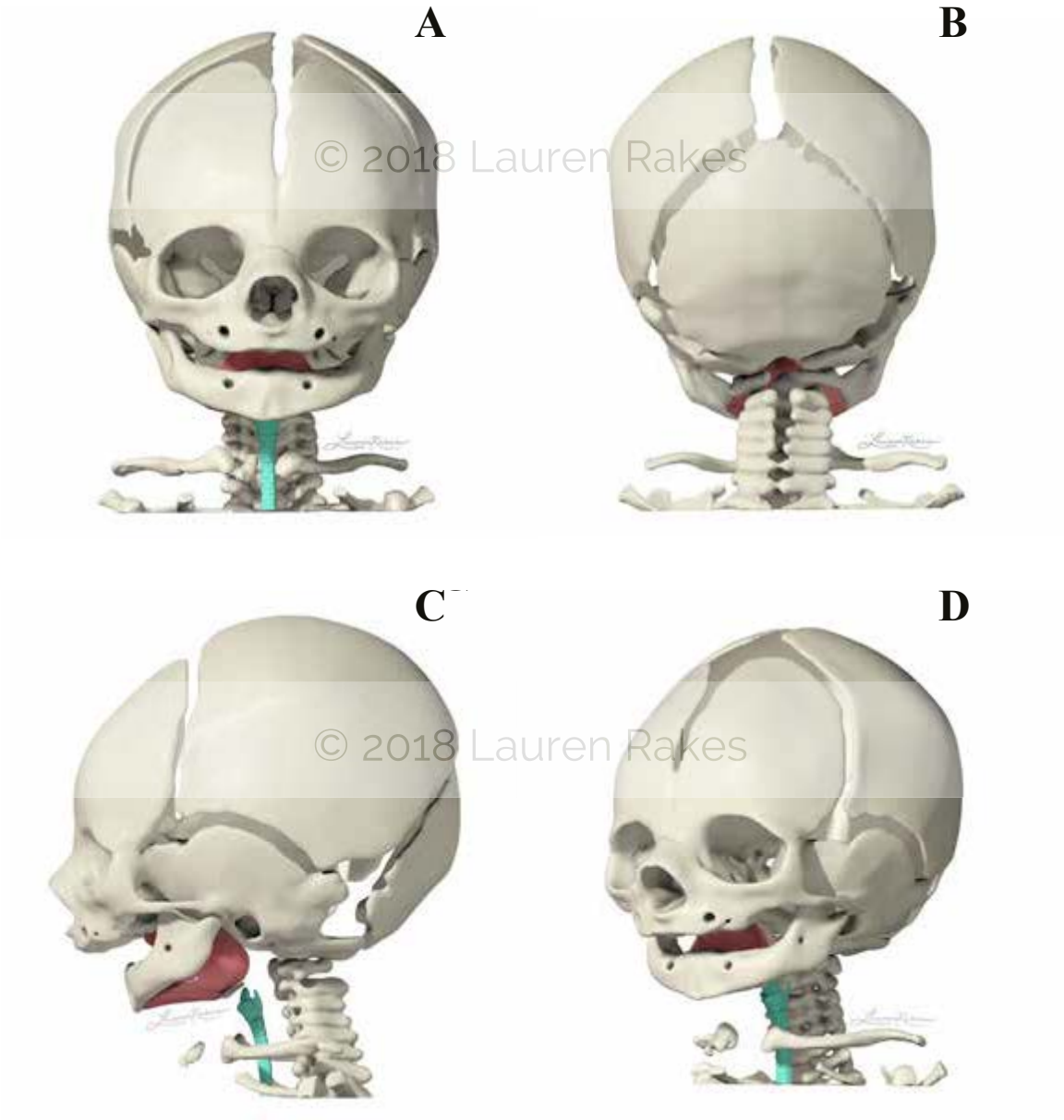


Figure 65. 3D model of Pierre Robin sequence infant skeleton. **(A)** Front. **(B)** Back. **(C)** Side. **(D)** 3/4 view.

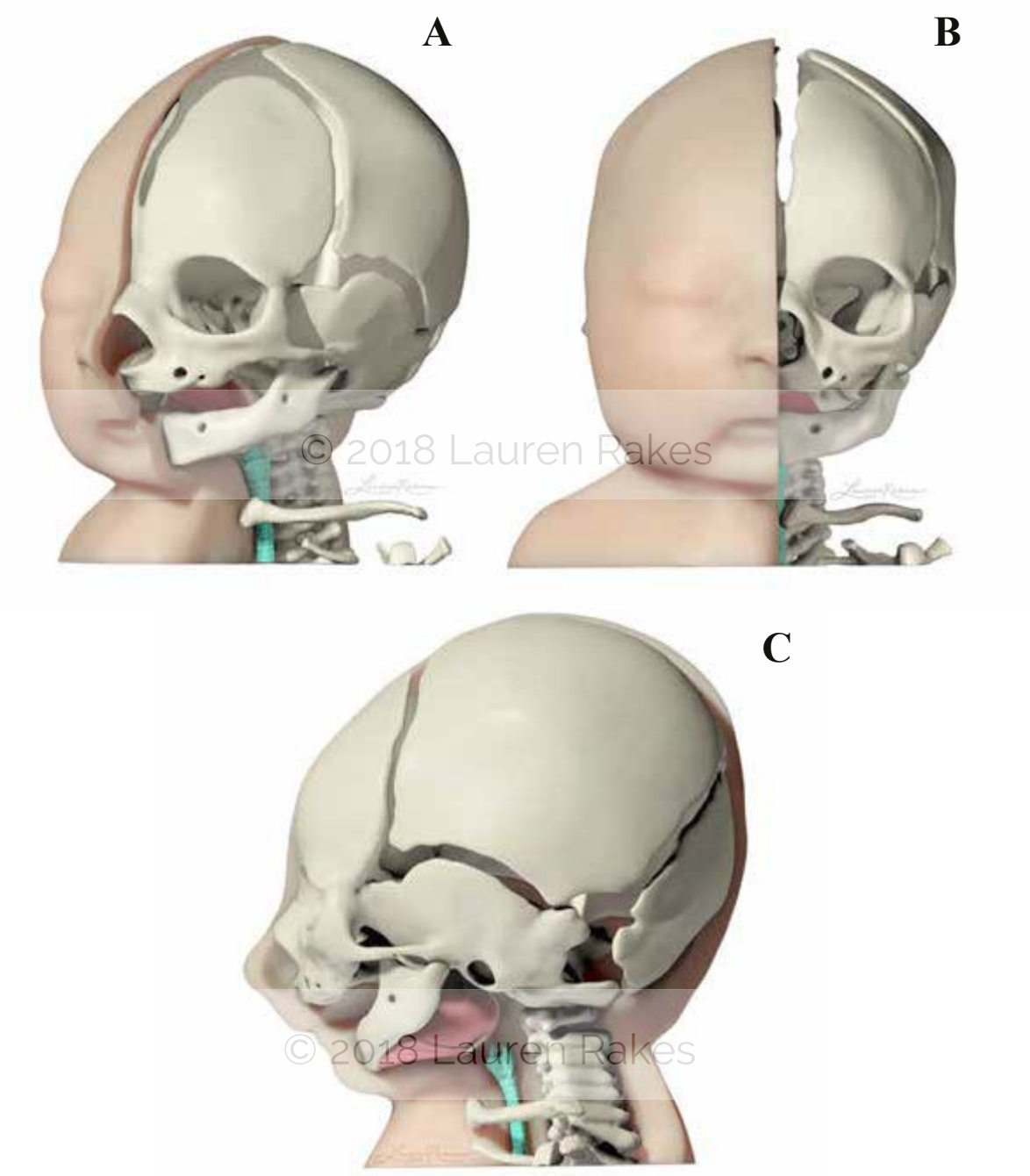


Figure 66. 3D model of Pierre Robin sequence infant with sagittal skin. **(A)** 3/4 view. **(B)** Front. (C) Side.

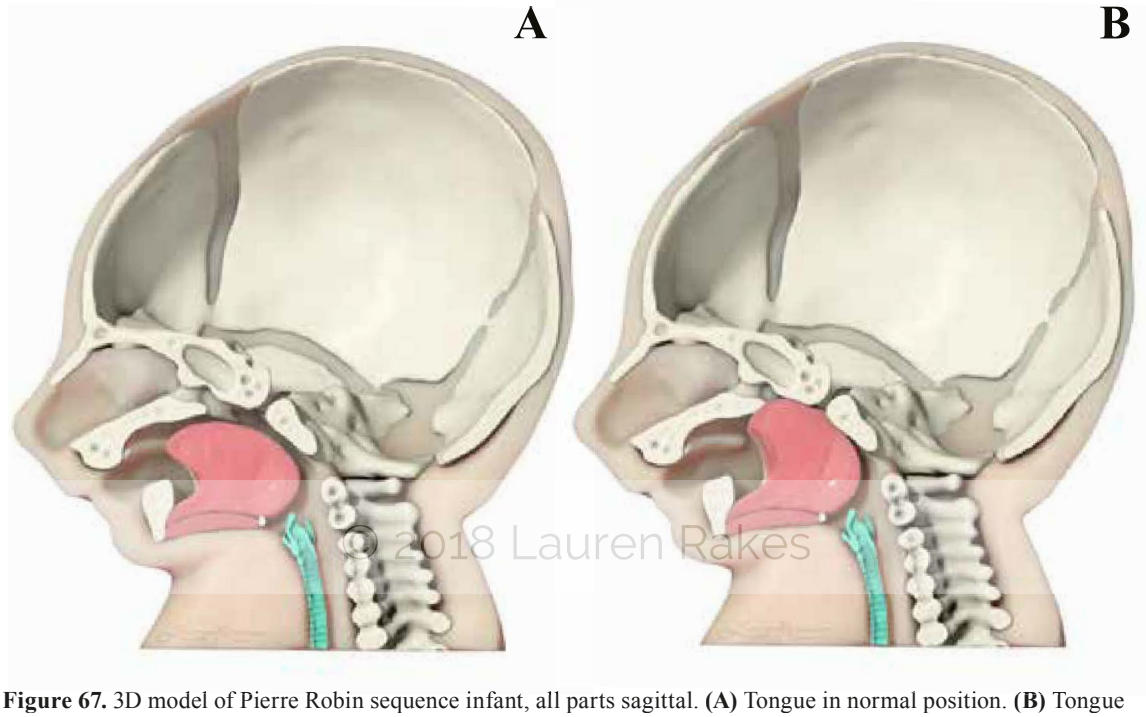


Figure 67. 3D model of Pierre Robin sequence infant, all parts sagittal. **(A)** Tongue in normal position. **(B)** Tongue positioning into cleft palate.

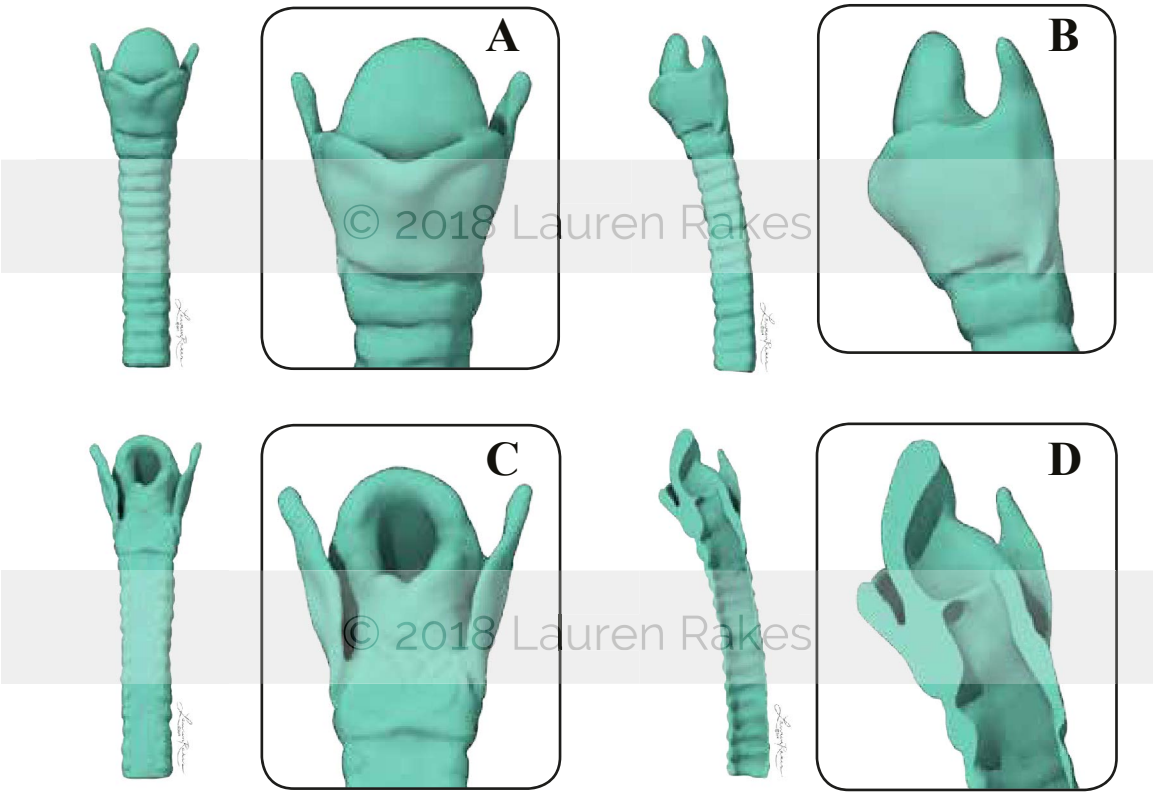


Figure 68. 3D model of larynx. **(A)** Front. **(B)** Side. **(C)** Back. **(D)** Sagittal, side.

Difficult Airway Algorithm for Micrognathic and Retrognathic Infants

A unique algorithm was developed specifically for micrognathic and retrognathic infants, using the adult Difficult Airway Algorithm as a basis. The algorithm consists of multiple airway management plans for intubating an infant with micrognathia (specifically Pierre Robin syndrome) and includes 39 unique steps along the pathway. The algorithm exists in a variety of formats, including an AI document (.ai) (Figure 69), a PDF, an online draw.io document, and a simplified interactive form within a WebGL application (Figure 76). The use of interactivity within the algorithm introduces a new way to learn and explore. The approach developed in this project may prove useful in creating similar treatment training algorithms for procedures related to conditions other than micrognathia/retrognathia.

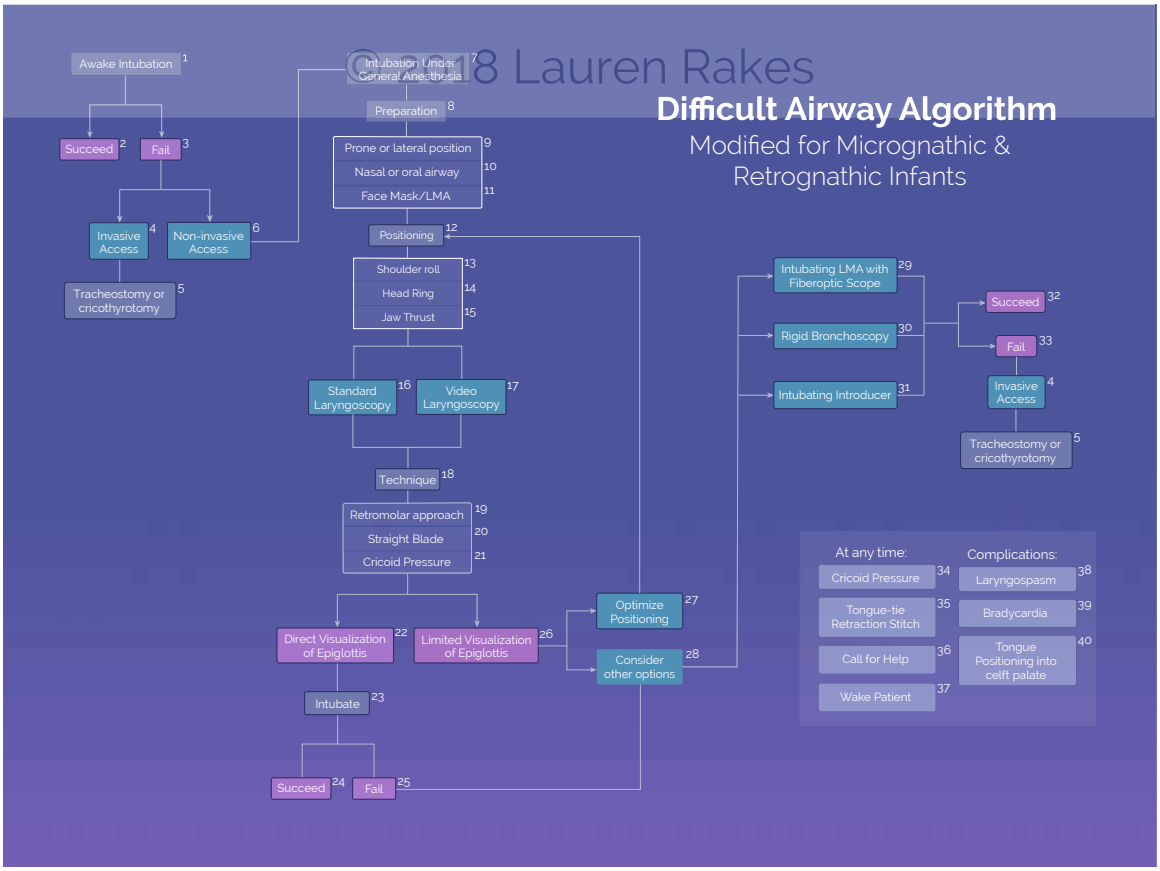


Figure 69. Difficult Airway Algorithm: Modified for Micrognathic and Retrognathic Infants *(refer to Figures 49-52 for legible text).*

Interactive WebGL Application

An interactive WebGL application was created in Unity. The application runs on any computer web browser but was thoroughly tested throughout this project in Google Chrome. The application opens on a **Home Screen**, where the user can choose to visit three sections: **3D Anatomy, Difficult Airway**

Algorithm, or **Background Information**. Additionally, the user can click on the **"info"** button in the upper right corner to learn about the application.

Figure 70. Web application Home Screen *(text not intended to be read).*

The 3D Anatomy section of the application features a 3D model of a Pierre Robin sequence infant, and a user interface that allows for certain parts of the model to turn on and off, fade, or switch to a sagittal cross-section. Users can rotate and pan the 3D model with the right and left buttons of a mouse respectively, as well as zoom with the mouse wheel. The model can be reset to its original position, or can snap to anatomical positions (anterior, posterior, left, and right) by clicking on one of the four buttons labeled "A" "P" "L" and "R". Users can return to the Home Screen by clicking on the **"home"** button, and can view instructions by clicking the **"question mark"** button.

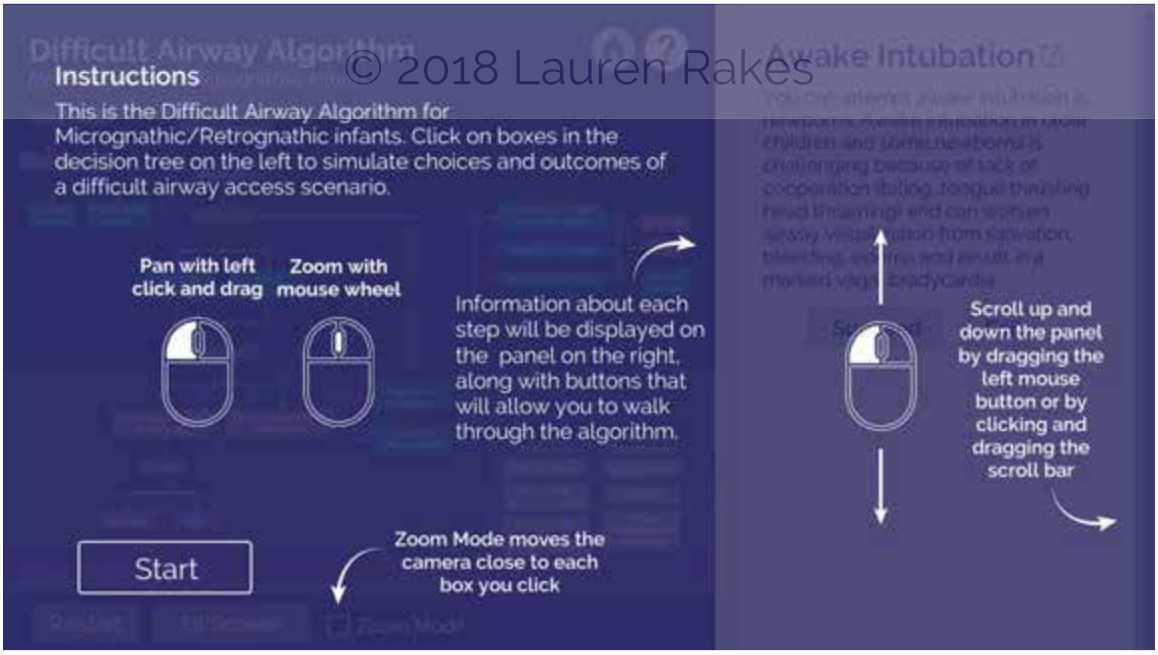
Figure 71. Web application 3D Anatomy Section: Instructions *(text not intended to be read).*

Figure 72. Web application 3D Anatomy Section *(text not intended to be read).*

Figure 73. Web application 3D Anatomy Section, skin transparent *(text not intended to be read).*

Figure 74. Web application 3D Anatomy Section, skin sagittal *(text not intended to be read).*

The Difficult Airway Algorithm section of the application has an area on the left that contains the "Difficult Airway Algorithm for Micrognathic & Retrognathic Infants" developed for this project, and an area on the right that holds an information panel. The users can explore the difficult airway algorithm by panning and zooming with the mouse, and can click "Fit Screen" to reset the algorithm to its original position. They can click on buttons in the algorithm to display more information about that particular step on the panel to the right. They can use the scroll bar or clicking and dragging to scroll up and down the panel on the right to see all the available information, and can click on the buttons below the content in the panel to progress along the algorithm. A toggle exists so the user can turn on and off "Zoom mode." "Zoom mode" moves the camera closer and centers it over each button on the algorithm as it is clicked. This feature is particularly useful when the application is running on a small screen.



© 2018 Lauren Rakes

Figure 75. Web application Difficult Airway Algorithm Section, instructions *(text not intended to be read).*

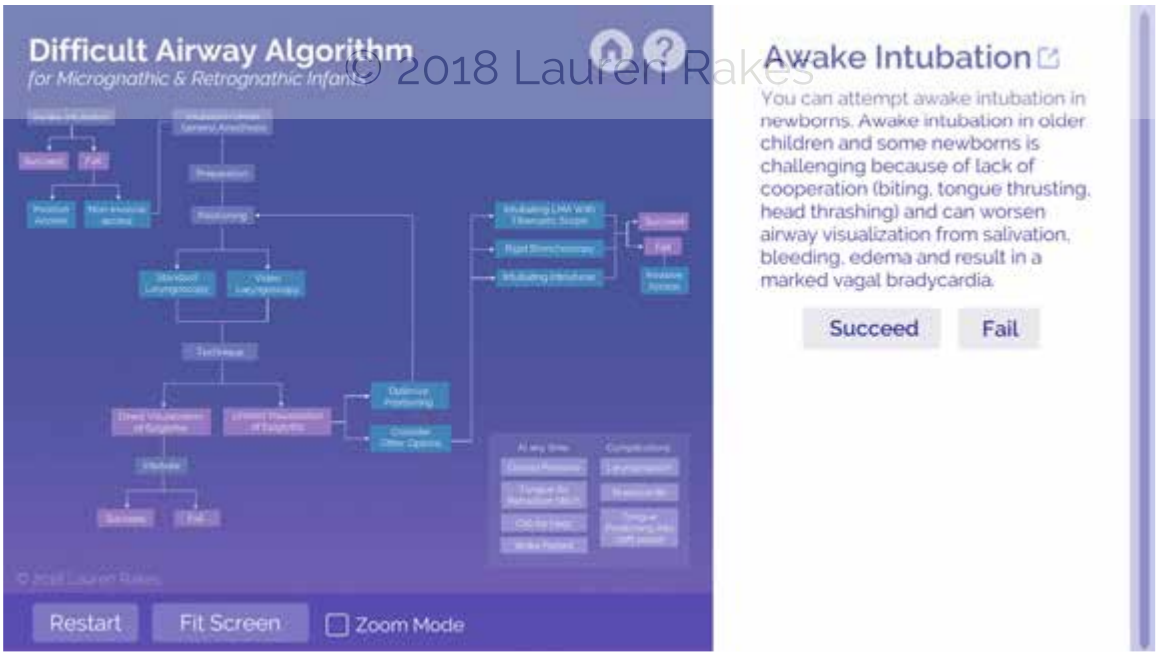


Figure 76. Web application Difficult Airway Algorithm Section, Awake Intubation *(text not intended to be read).*

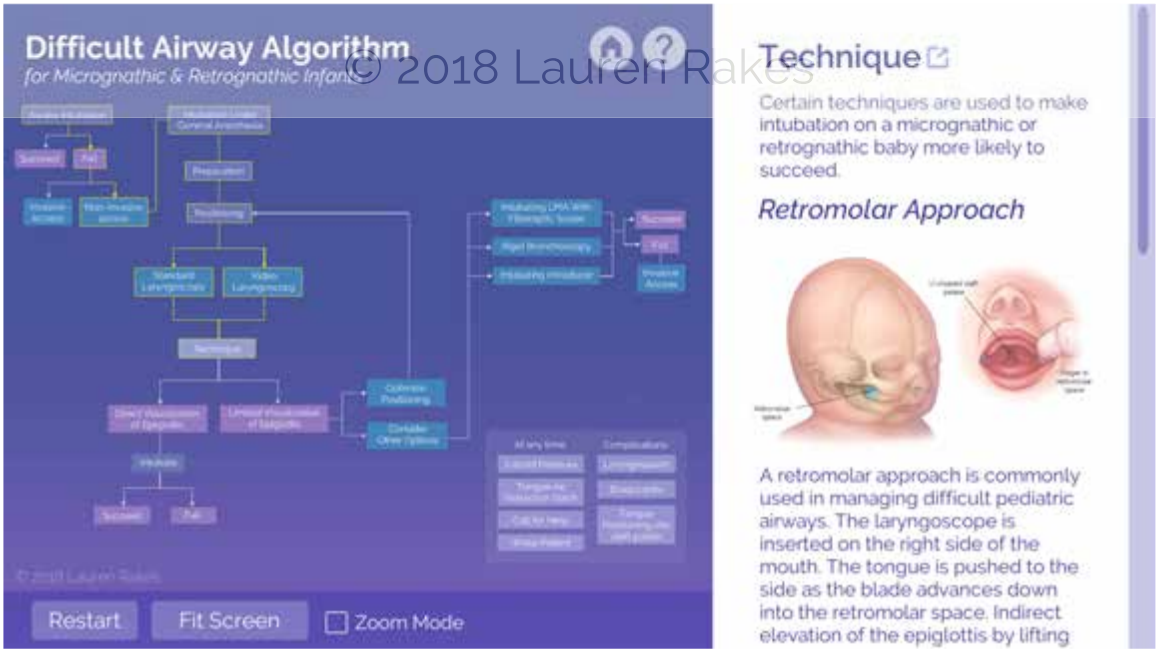


Figure 77. Web application Difficult Airway Algorithm Section, Technique *(text not intended to be read).*

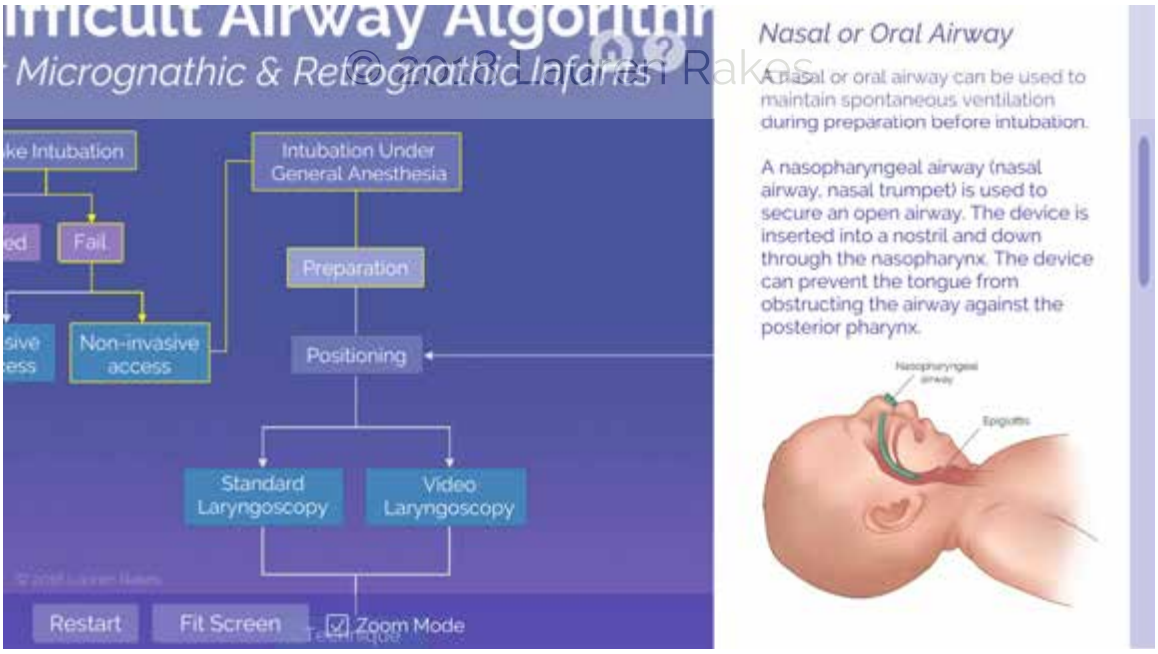


Figure 78. Web application Difficult Airway Algorithm Section, Nasal or Oral Airway *(text not intended to be read).*

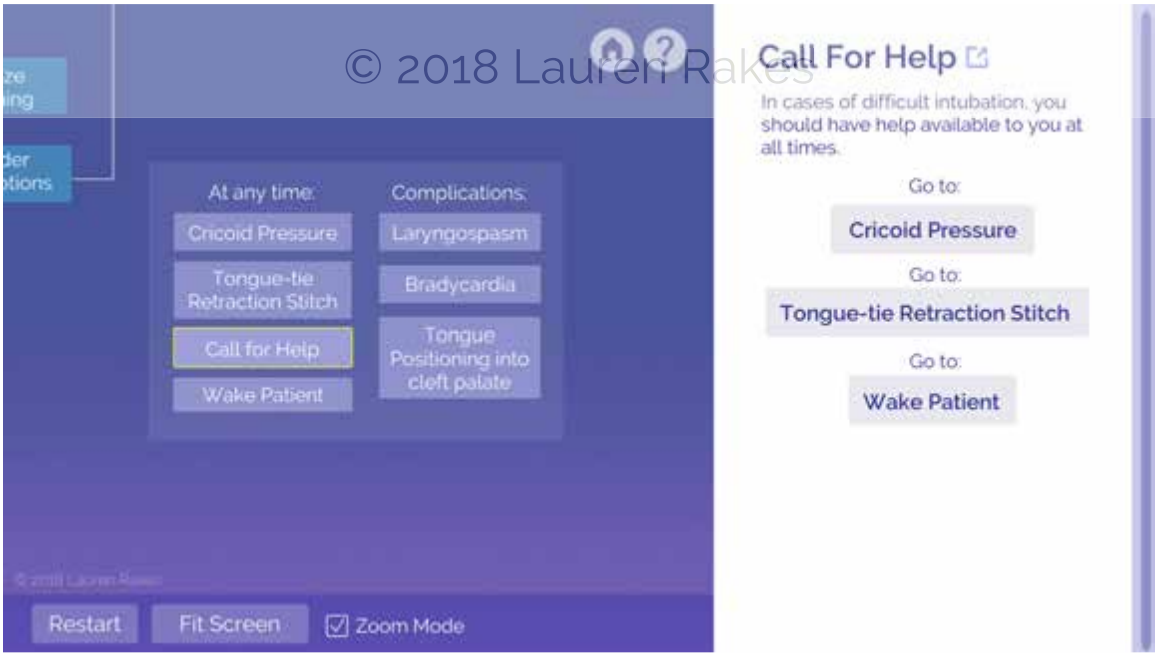


Figure 79. Web application Difficult Airway Algorithm Section, Call for Help *(text not intended to be read).*

The Background Information section of the application features a simple left side menu bar with different topics for the user to explore, including “About Micrognathia and Retrognathia,” “Pediatric Anatomy,” “Management,” “Surgery,” and “Complications”. Information about each topic is displayed on a panel to the right, along with explanatory illustrations. Buttons can be clicked next to each header in the panel to open an online PDF in a new tab. The PDF (containing all the information inside the Background Information section as well as the Difficult Airway Algorithm Section) will open to the page with the corresponding information. The user can view the figures more closely on the online PDF, print it out for future reference, or even click on buttons to play short animations that open in a separate tab.

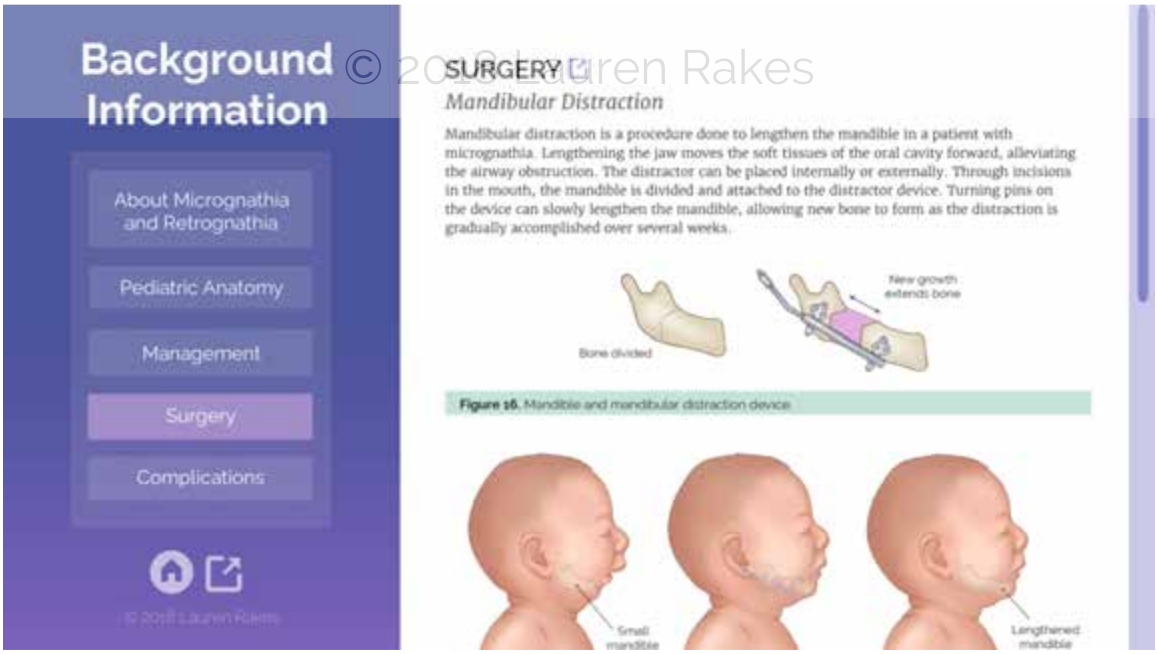


Figure 80. Web application Background Information Section *(text not intended to be read).*

2D Artwork

30 illustrations were created for use this project. Combined with the interactive elements in the application, the illustrations demonstrate and explain the steps along the decision-tree of the “Difficult Airway Algorithm for Micrognathic & Retrognathic Infants”. The illustrations are effective didactic aids that help the users understand pediatric and micrognathic anatomy, as well as basic airway management techniques.

Figure 81. Pierre Robin sequence infant.

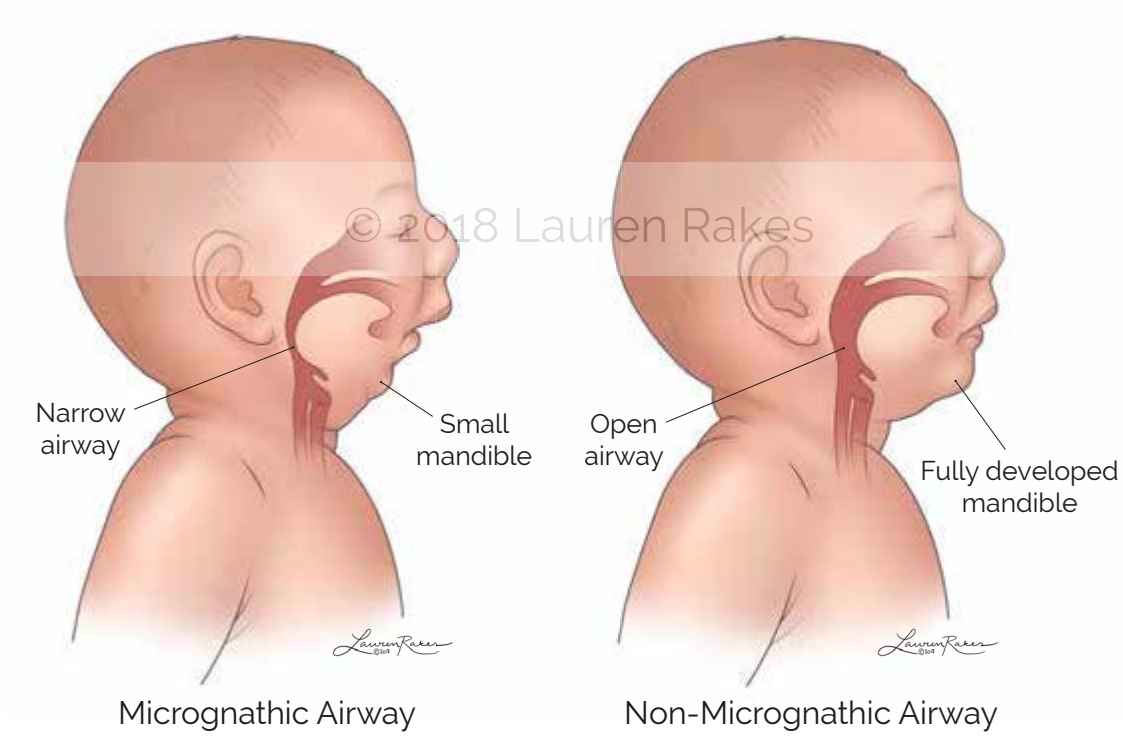


Figure 82. Micrognathic vs non-micrognathic airway.

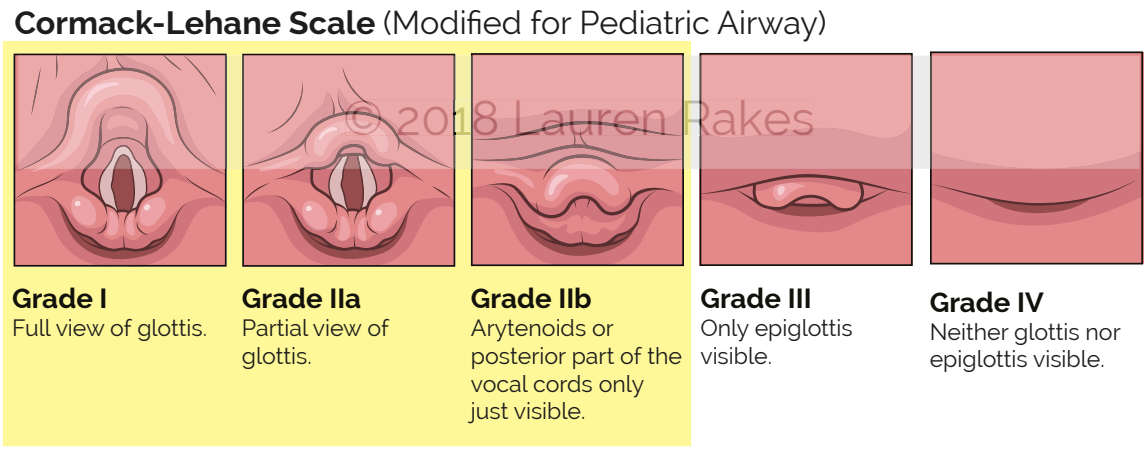


Figure 83. Cormack-Lehane scale, Grades I, and II.

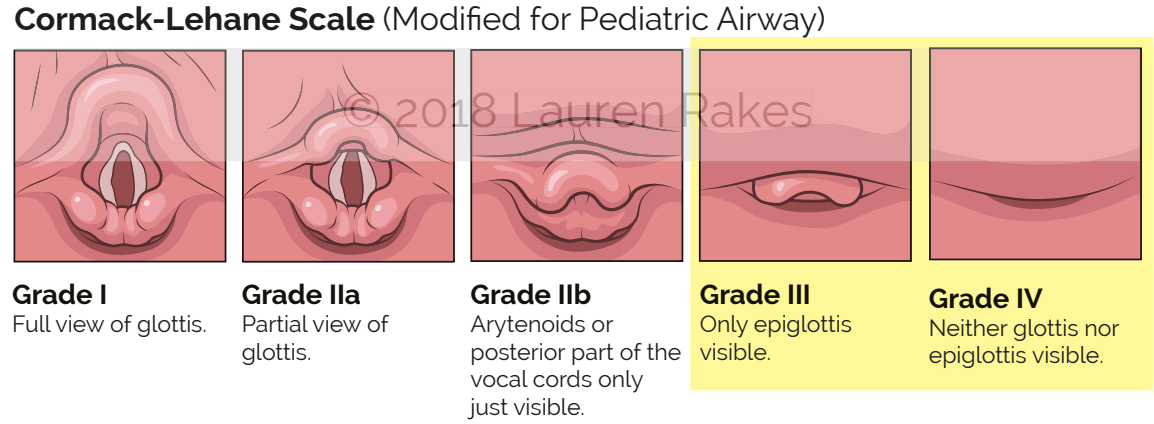


Figure 84. Cormack-Lehane scale, Grades III, and IV.

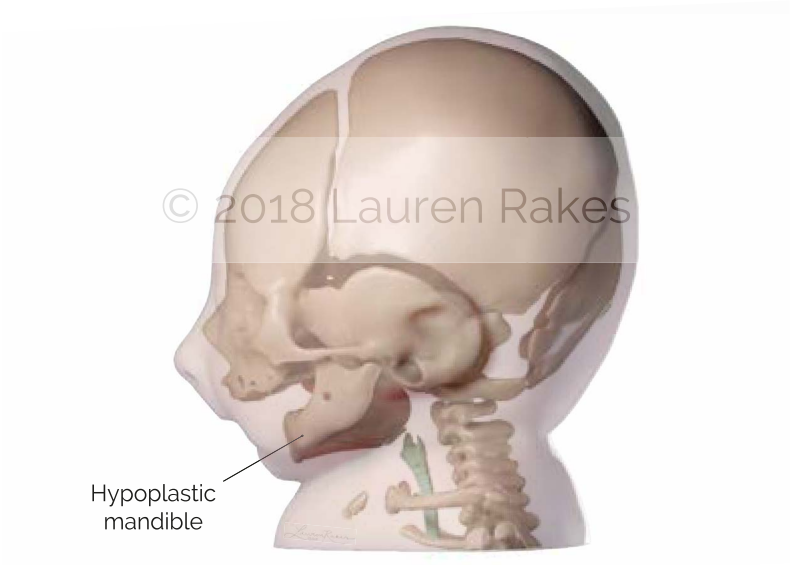


Figure 85. Hypoplastic mandible.

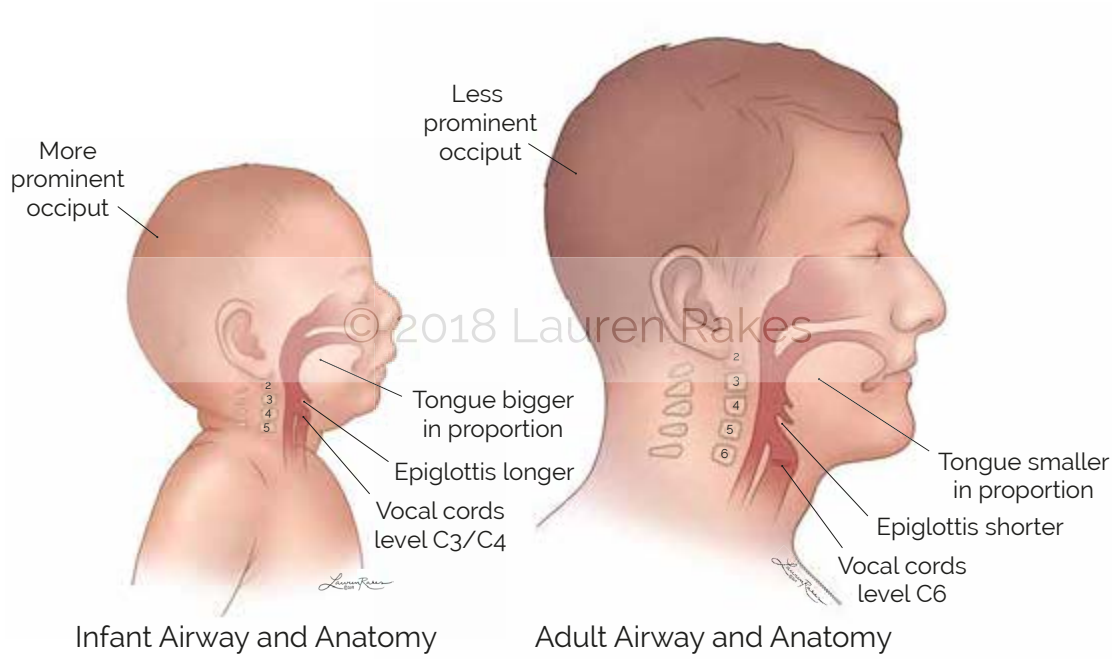


Figure 86. Infant vs adult airway anatomy.

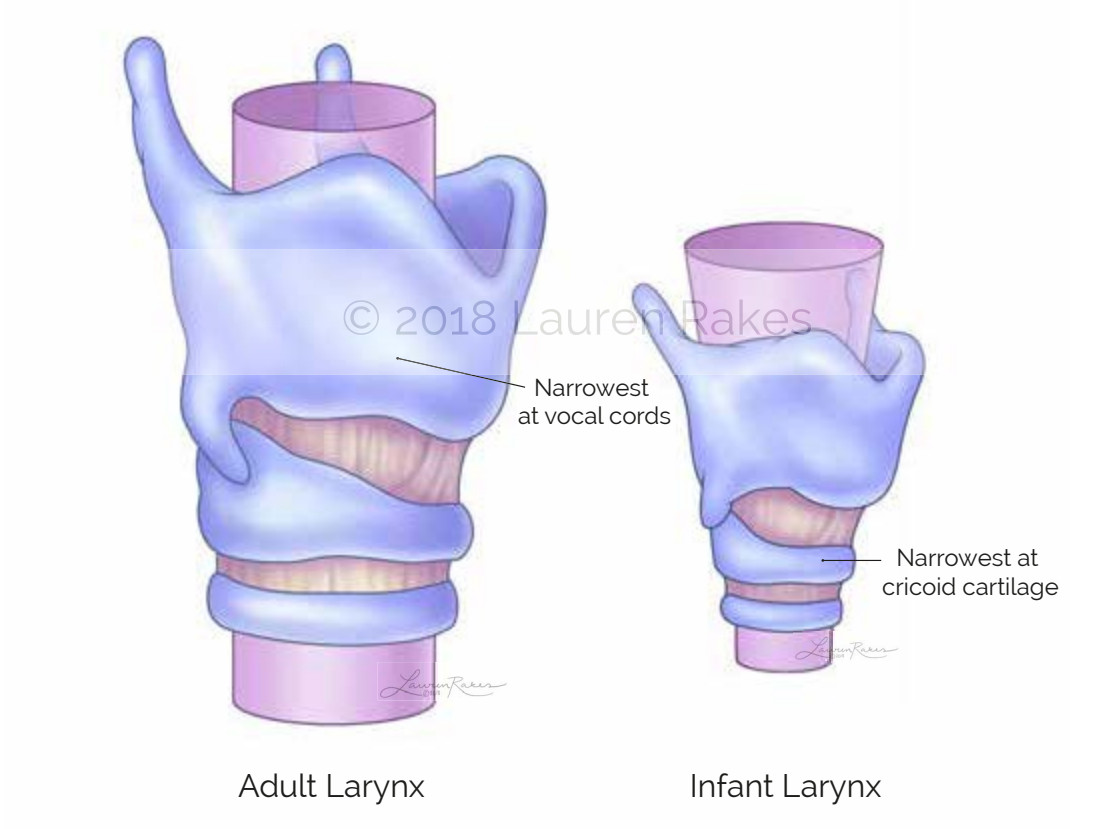


Figure 87. Infant vs adult larynx.

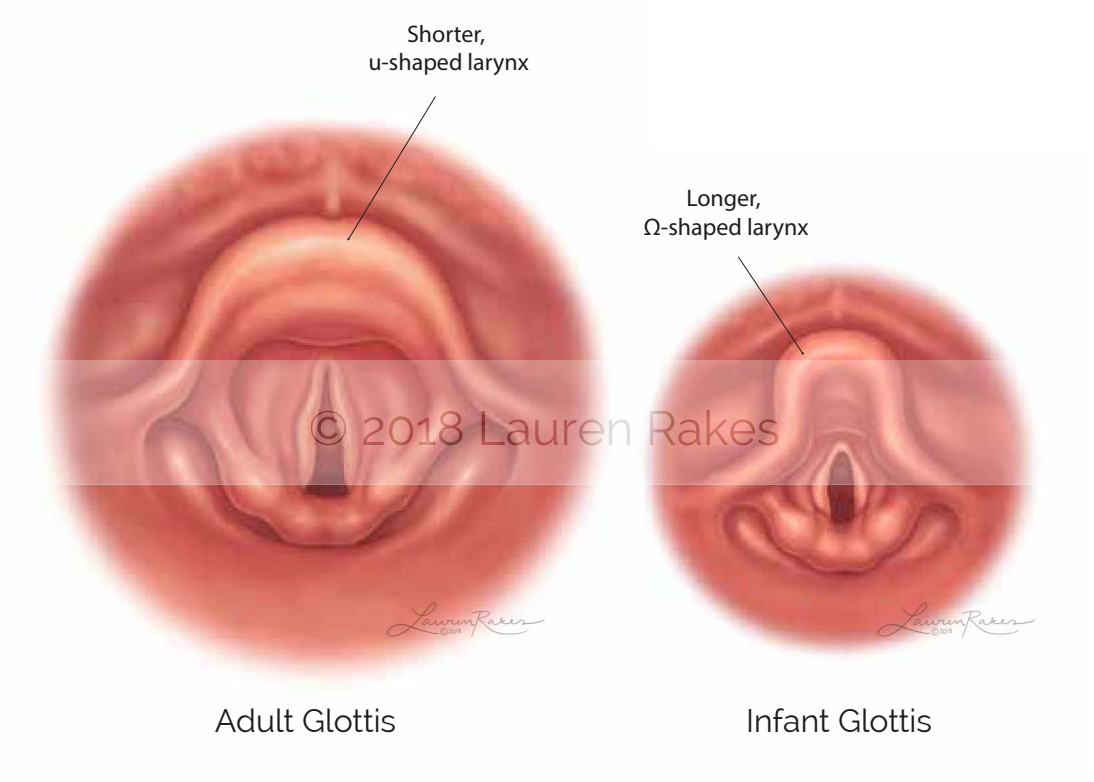


Figure 88. Infant vs adult glottis.

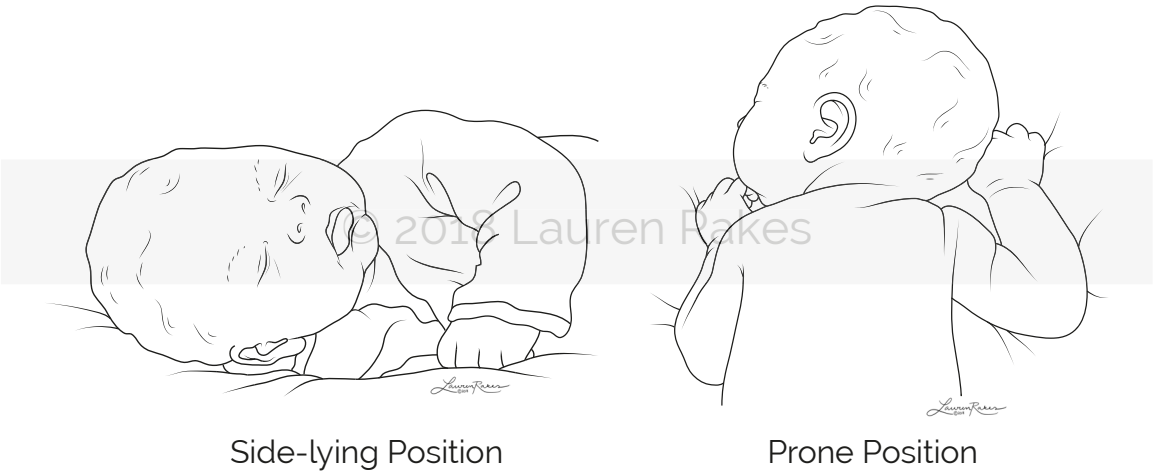


Figure 89. Infant positioning.

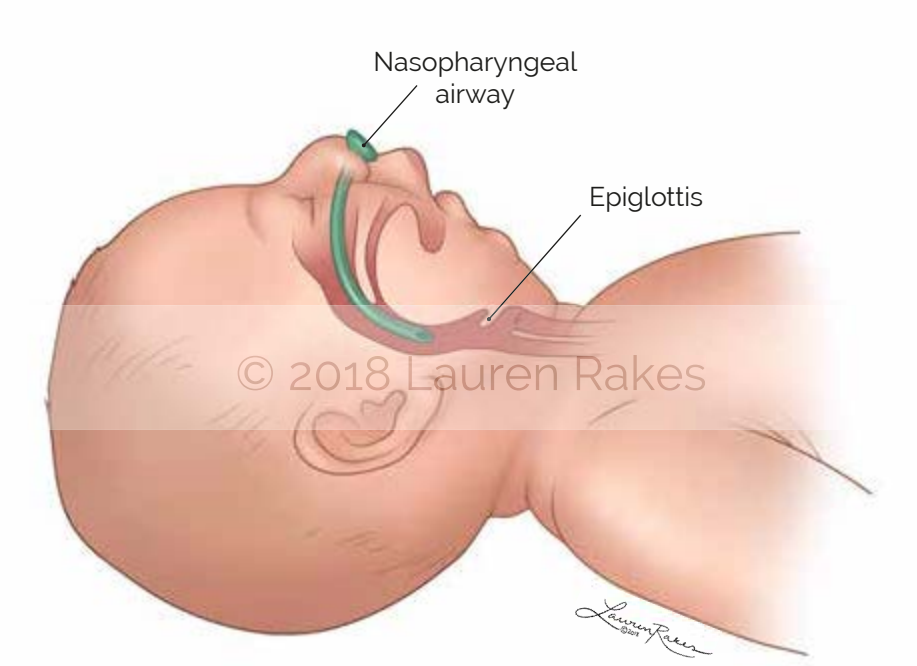


Figure 90. Nasopharyngeal airway.

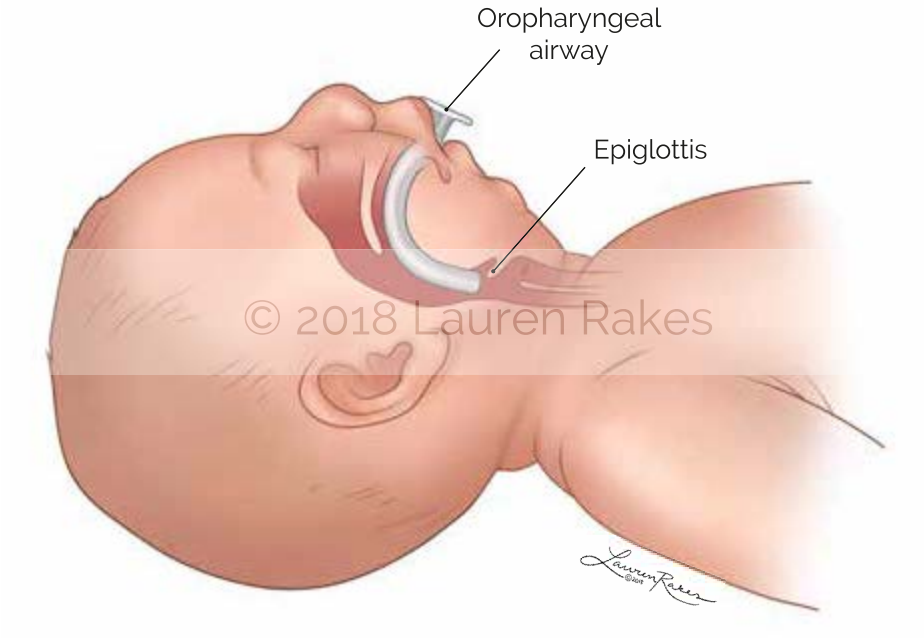


Figure 91. Oropharyngeal airway.

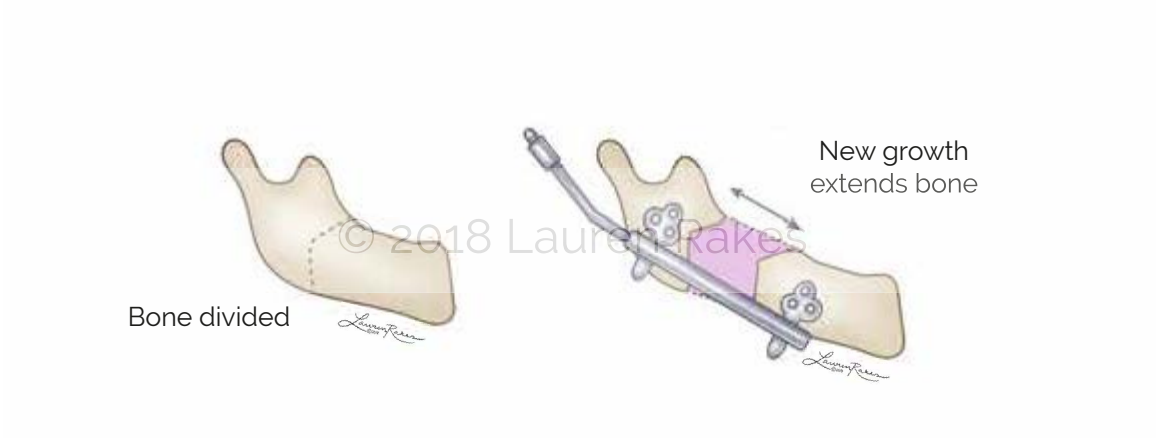


Figure 92. Mandibular distraction device.

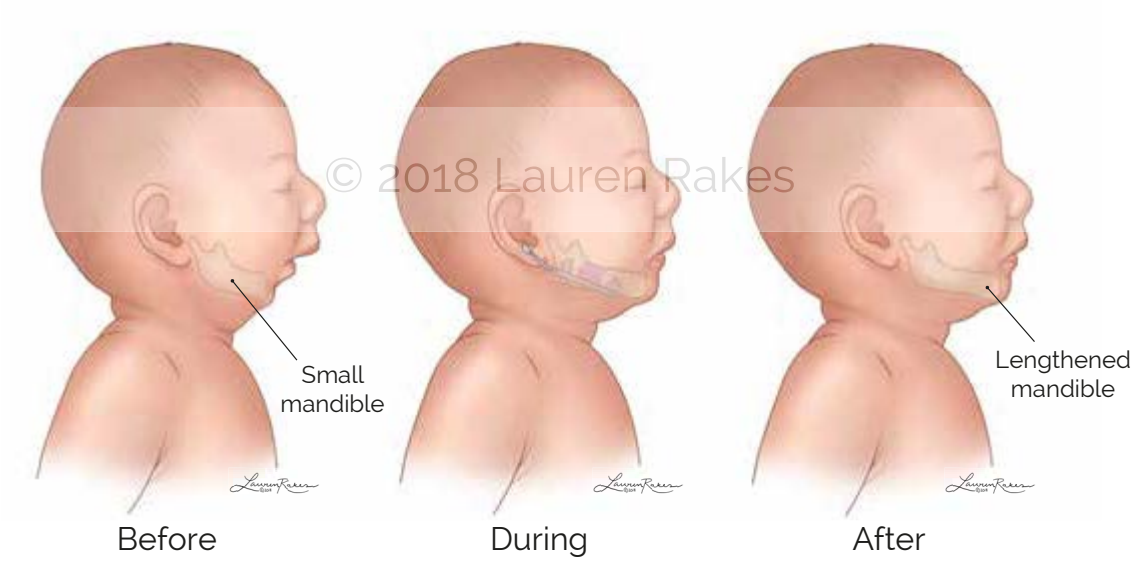


Figure 93. Mandibular distraction.

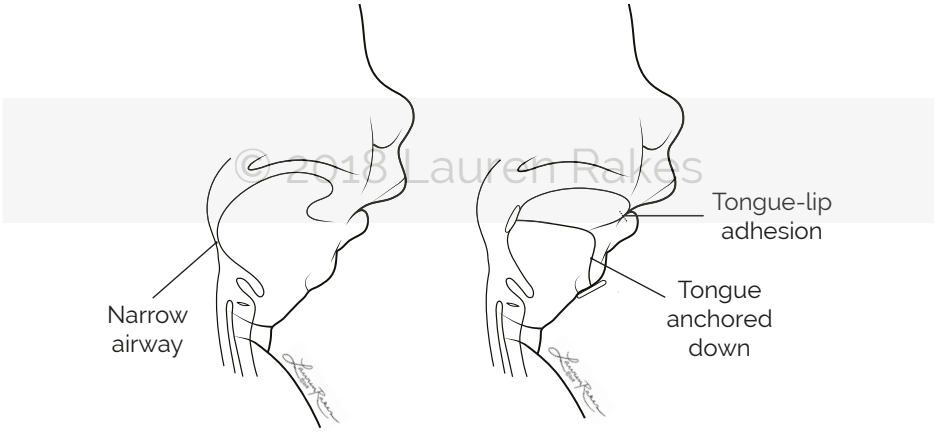


Figure 94. Tongue-lip adhesion.

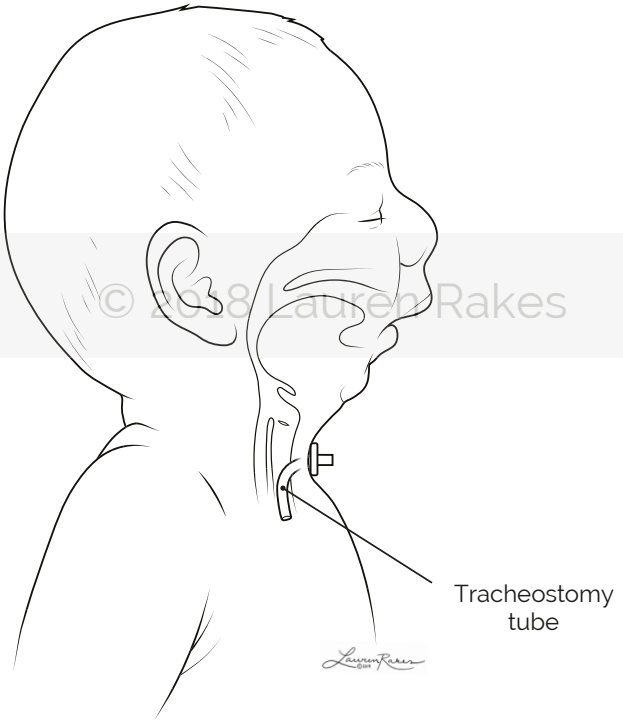


Figure 95. Tracheostomy.

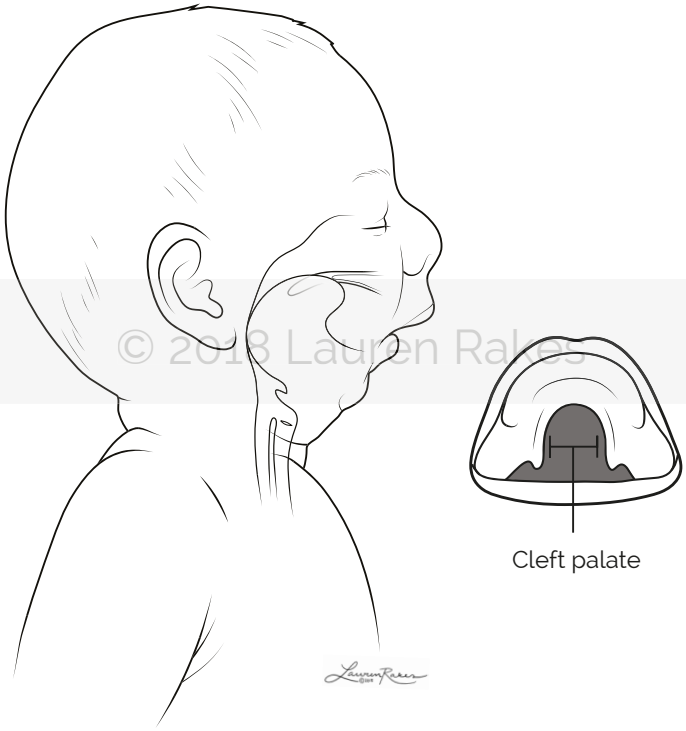


Figure 96. Tongue positioning into cleft palate.

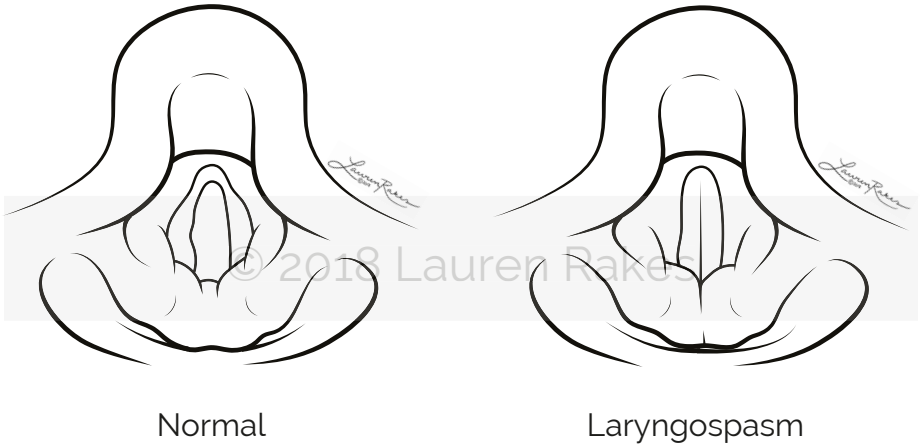


Figure 97. Laryngospasm.

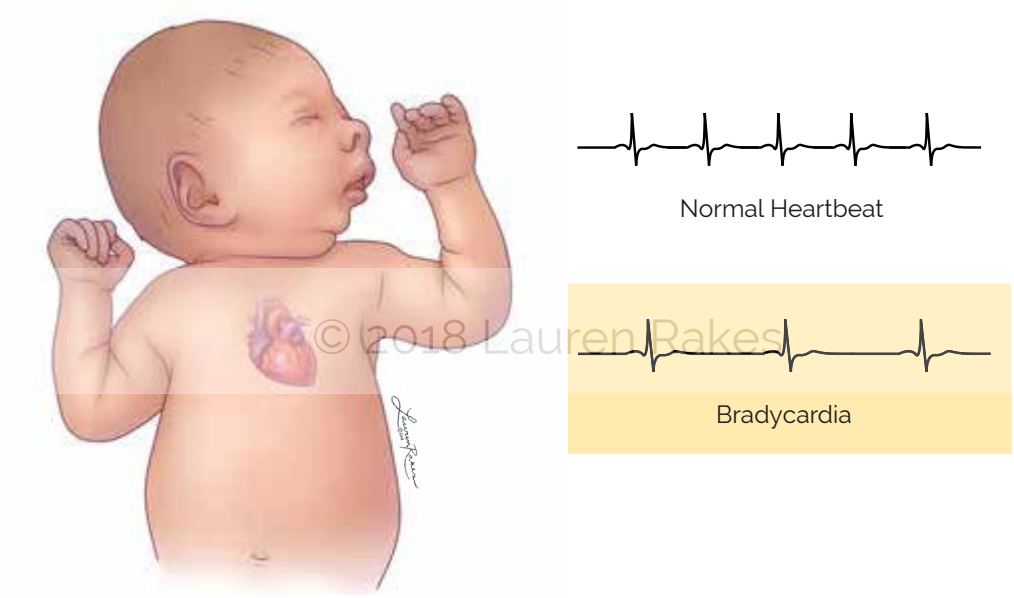


Figure 98. Bradycardia.

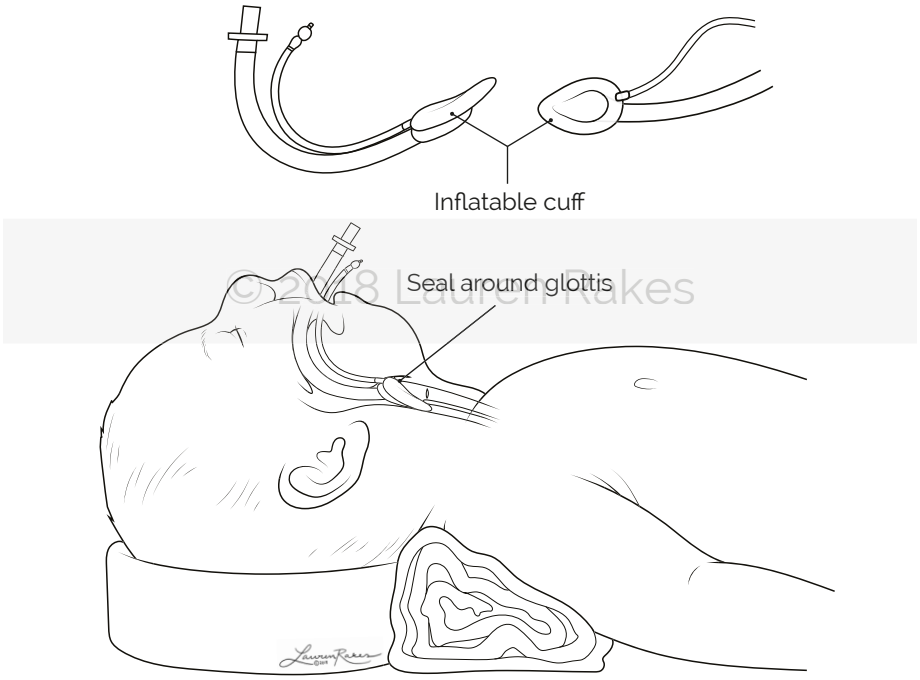
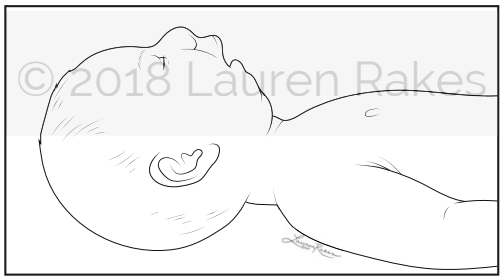
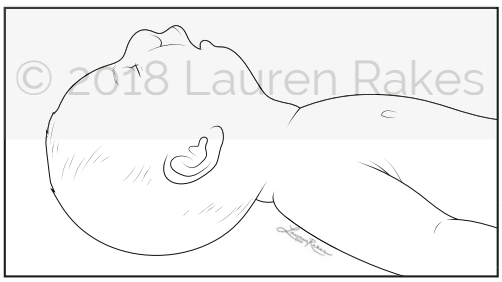


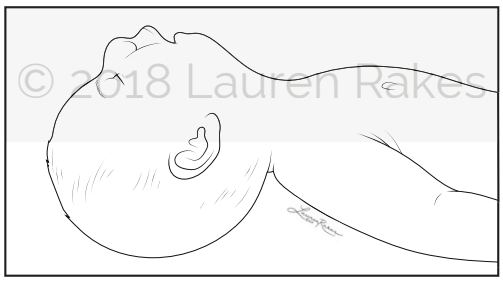
Figure 99. Laryngeal mask airway.



Incorrect positioning:
Atlanto-occipital joint **flexed**, lower cerivical joint **flexed**.



Correct positioning:
Atlanto-occipital joint **extended**, lower cerivical joint **flexed**.



Incorrect positioning:
Atlanto-occipital joint **extended**, lower cerivical joint **extended**.

Figure 100. Correct positioning for intubation.

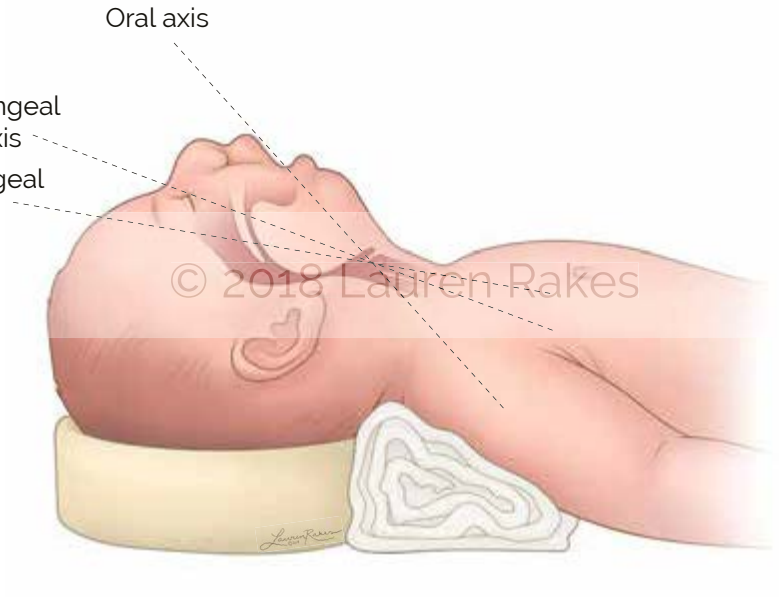


Figure 101. Axes of alignment.

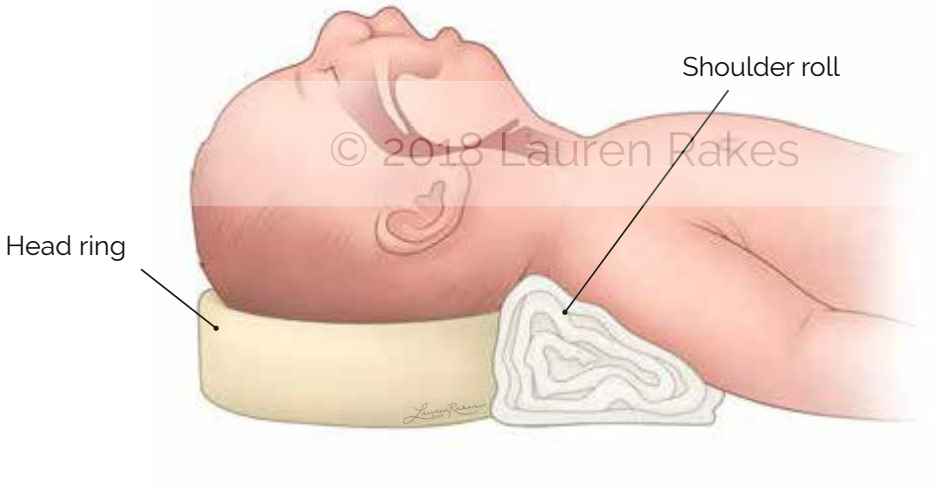


Figure 102. Head ring and shoulder roll.

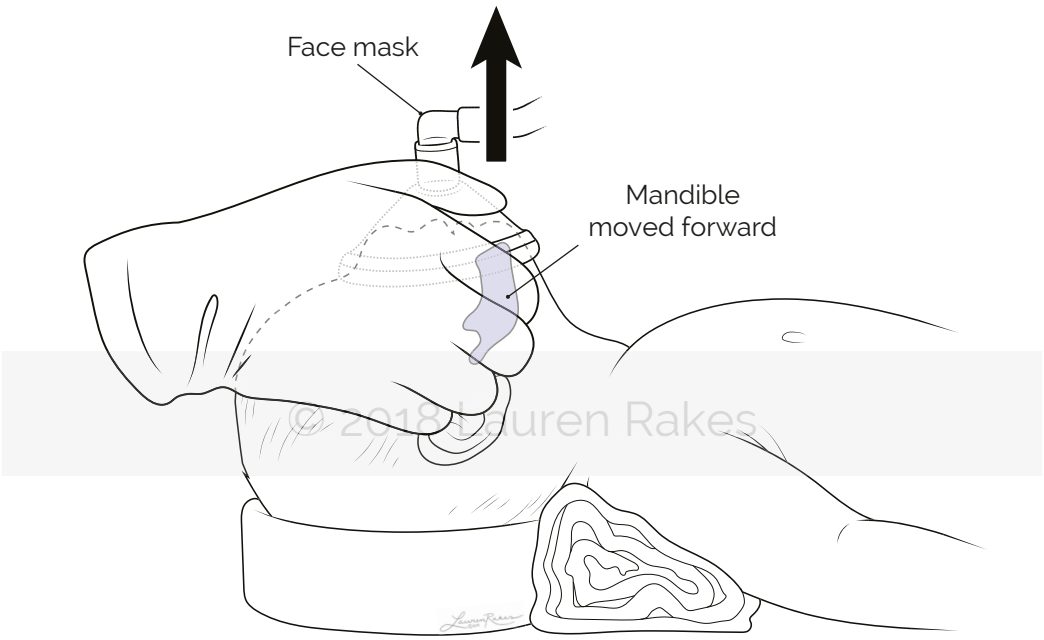


Figure 103. Jaw thrust.

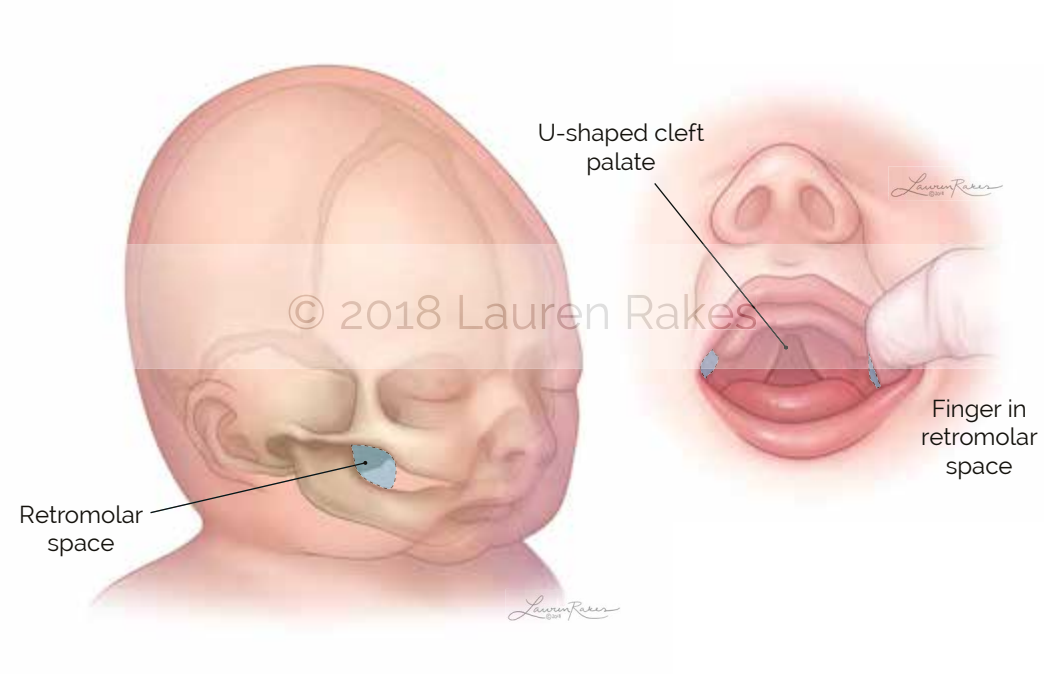


Figure 104. Retromolar space.

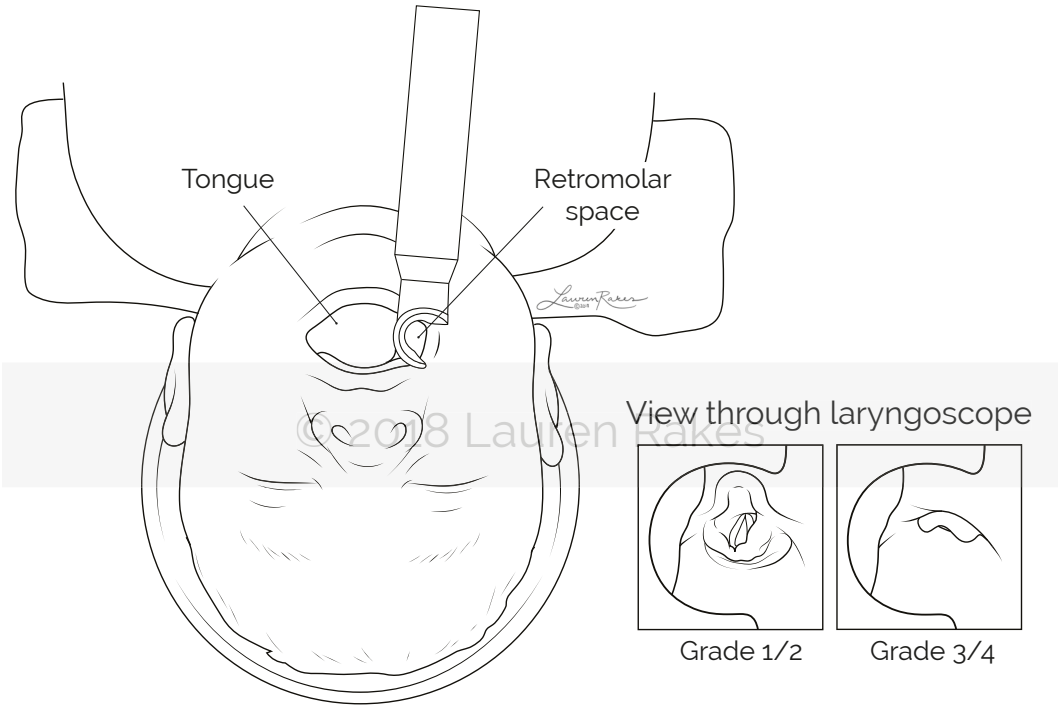


Figure 105. Retromolar intubation.

Figure 106. Cricoid pressure.

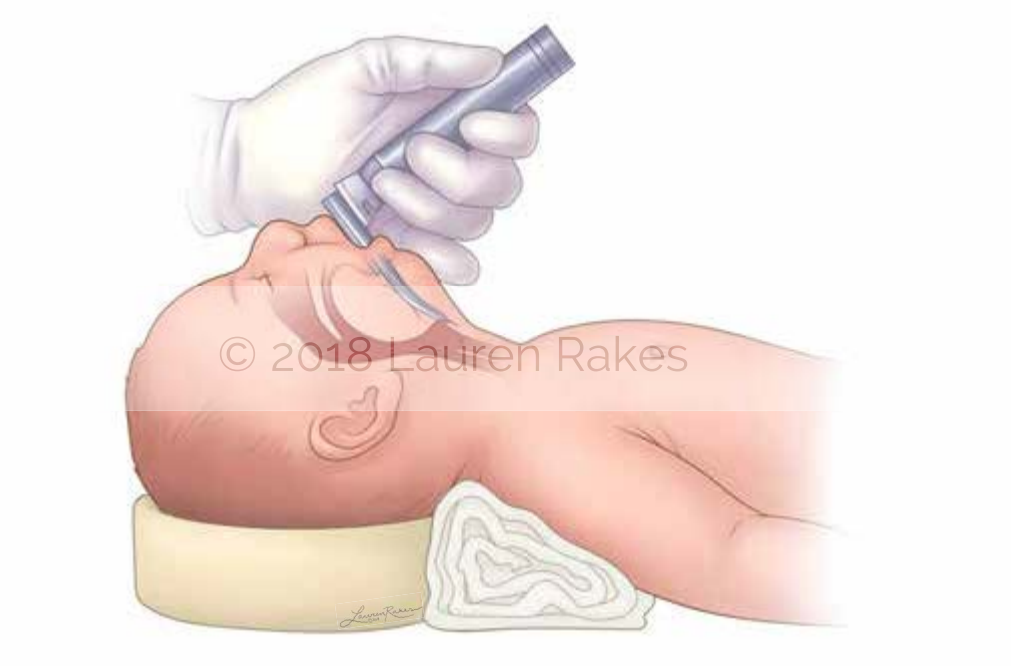
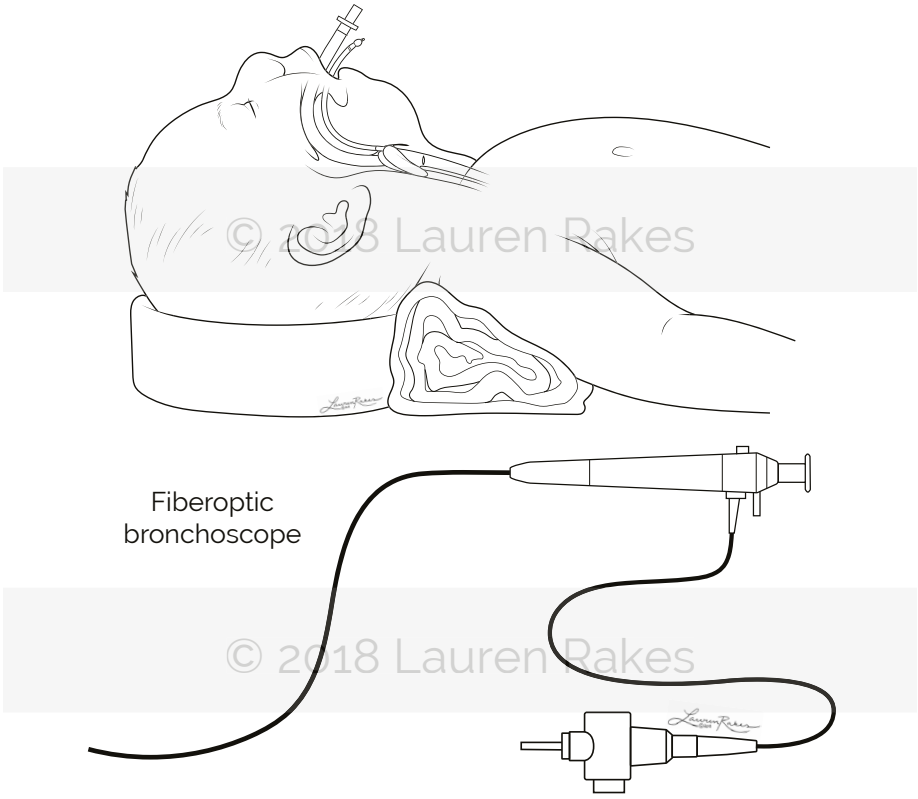

© 2018 Lauren Rakes

Figure 107. Intubation.



© 2018 Lauren Rakes
© 2018 Lauren Rakes

Figure 108. LMA with fiberoptic bronchoscope.

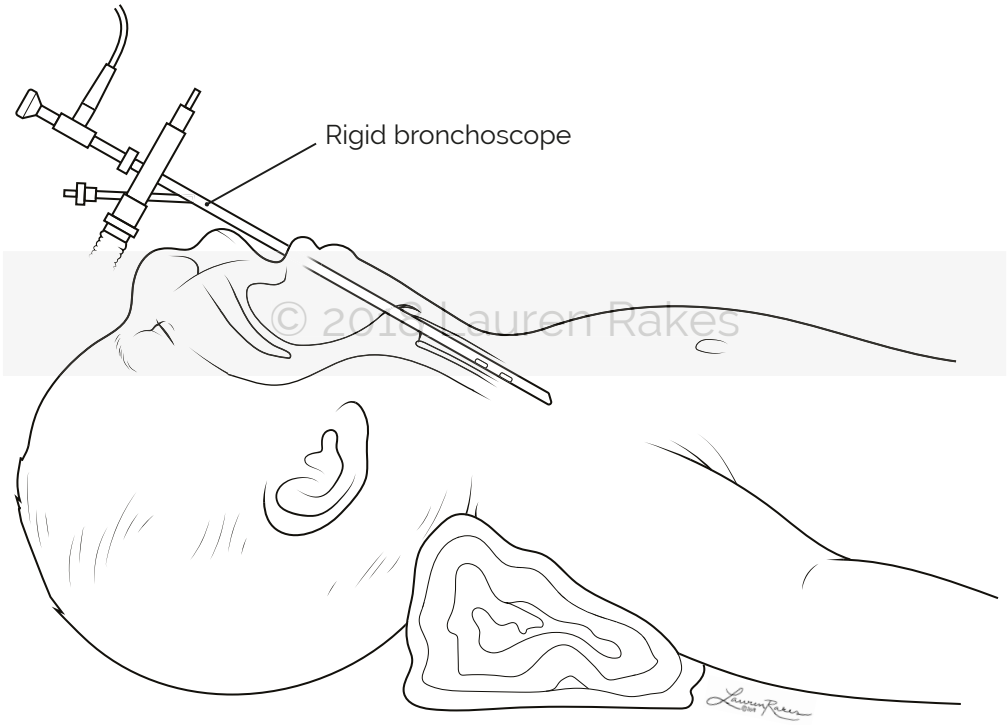


Figure 109. Rigid bronchoscope.

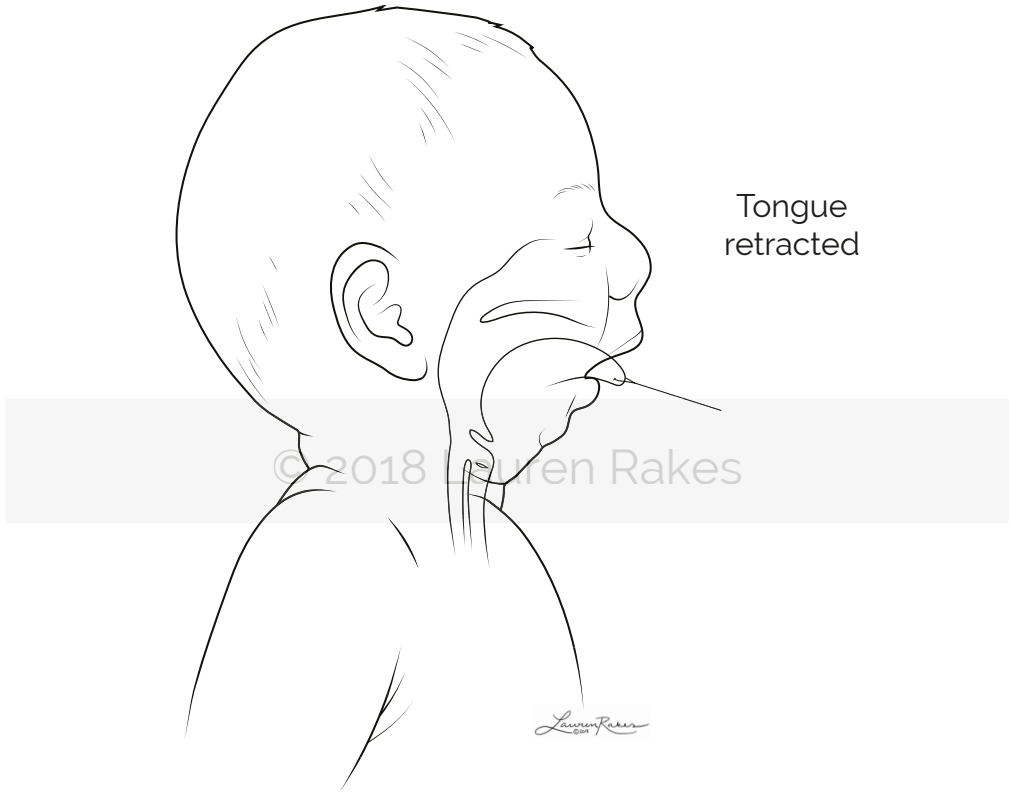


Figure 110. Tongue retracted.

2D Animations

Four animations were created for use in the interactive application. Users can view the animations by clicking the "Play video" buttons next to certain figures in the online PDF.

The first animation demonstrates the technique of retromolar intubation of a child with micrognathia, including positioning and airway anatomy.

The second animation demonstrates the technique of using cricoid pressure to move the larynx into a better position for visualization and intubation.

The third animation depicts the axes of intubation the physician uses to ensure the patient is positioned optimally for visualization and intubation. The oral, nasopharyngeal, and laryngeal axis must be aligned properly for a successful and safe intubation.

The final animation addresses bradycardia, a condition in which there is an abnormally slow heartbeat. Bradycardia can be a complication of airway management, especially in a pediatric difficult airway case. The animation shows the rate of a healthy infant's heart compared to one suffering from bradycardia.

PDF

A comprehensive online interactive PDF containing all the text and images from the application was created for the user to review during or outside of application use. The user can access the PDF by clicking on "New tab" symbols inside the application. "Play video" buttons in the PDF open a new tab and show the user animations of certain figures.

Booklet

Four full-color printed booklets based on the contents of the PDF will be supplied to the Johns Hopkins Hospital Department of Pediatric Anesthesiology to assist in teaching physicians about micrognathia airway management (Figure 111). Future plans include making the booklet available to healthcare providers outside of Johns Hopkins Hospital.

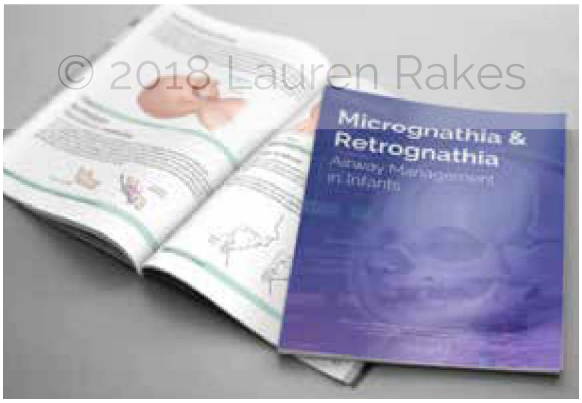


Figure 111. Mockup of PDF booklets *(text not inded to be read).*

Access to Assets Resulting from this Thesis

Assets resulting from the work of this thesis can be partially found at **www.laurenrakes.com** or by contacting the author at **lprakes@gmail.com**. The author may also be reached through the Department of Art as Applied to Medicine via the website **medicalart.johnshopkins.edu.**

DISCUSSION

Project Goal

The primary goal of this project was to build an application to enhance the decision-making of pediatric anesthesiologists and otolaryngologists during management of the pediatric micrognathic airway. This application was designed to promote the safe treatment of infants with micrognathia and increase the general awareness of the dangers of intubation of these difficult airways without adequate preparation.

Innovations

Interactivity

The use of interactivity within a flowchart has been used for both education and entertainment. The website Bustle offers entertaining flowcharts that depict a variety of different simulations, including "Would you survive the Oregon Trail?" and "What 90s Pop Star are You?" These games rely on the user to choose from various decisions to advance through a flowchart until an end result is reached.

In medical education, interactivity has been used to simulate medical scenarios in a safe environment without risk to patients. In 2013, Medical Faculties Network created a study to examine the effectiveness of interactive algorithms for teaching and learning acute-medical procedures. The study showed a positive reaction from students using algorithms to learn critical care medicine. The student participants expressed desire to implement this form of learning in their normal school education.

This project implements interactivity within an algorithm in a unique way. The users can learn about each step of the algorithm and understand the reasoning behind each decision, not just simulate choices without any further explanation. In this way, the users can better grasp the pathways of the entire algorithm.

Subject matter

The algorithm created for this web application offers a way to learn how to manage micrognathia, a specific type of difficult pediatric airway. Currently there are not many publicly available resources of any kind focusing on the management of micrognathia. While many aspects of all pediatric difficult airways are consistant among different conditions, it is important for a physician to understand the qualities that make each condition unique. This project creates a foundation for the expansion of learning resources and tools that can help improve the quality of life for micrognathic infants.

Accessibility

The integration of the flowchart within a web application creates an accessible way to reinforce learning in the users' leisure time. The users only need to open a link on their browser to access the web application, and do not need to download a separate program or application. A PDF of the information available in the application can also be printed to review without a computer.

Considerations

Reference Materials

There are a lack of resources showing the anatomy for micrognathic children. The DICOM CT data set provided for this project was useful as a base for the 3D model, and as reference for illustrations, but there were limited other resources for visualizing the airway of a child with micrognathia. The subject is not well addressed in existing illustrations or animations, and is generally only described in text in journals and textbooks.

Additionally, the incidence of children with micrognathia is relatively low. Johns Hopkins Hospital only sees one or two infants with micrognathia every couple of months. Because of this, only one CT data set was available at the start of this project. Having other CT data sets would be beneficial for the expansion of this project, so that the anatomy could be averaged and a model of a typical patient with Pierre Robin sequence could be created.

Exploratory Approach to Design

Initial designs, thumbnails, and sketches were made during the development of this project, detailing how the application and its various sections would appear and how the user would interact with the various elements within. Due to the nature of working with new software, these designs morphed over time. Technology, software, and time limitations kept the project from expanding too far. The road blocks and successes with each software impacted the design of the application so that some features were eliminated from the initial concept and some were added. In the end, the design of the overall application was based on initial sketches, user testing throughout the process of development, and input from content advisors and preceptors.

This exploratory approach to design resulted in an application with sections that had different layouts, however great care was taken to maintain consistency throughout the project of the color palettes, text, and treatments of images. Thus, the aesthetic and general design of the application was cohesive from start to finish.

Limitations

A person intending to replicate parts of this project or adapt it for a new application should consider their current knowledge, ability, and time available to learn programs such as Unity and programming languages like C#. Many free resources are available for learning these new technologies online, however one must consider the time required to effectively implement what they have learned.

It was discovered during this project that the free version of Unity does not support movie clips. This meant that the animations created for this project could not be played within the application. This problem was solved by creating buttons that were hyperlinked to the animations on the online PDF and therefore could be opened via that path. While this solution was effective, it is not as desirable as having the animation incorporated directly into the program.

Further Application Development

The application currently focuses only on micrognathic infants, and even more specifically on the particular anatomy of Pierre Robin sequence infant. In future iterations of this application, other conditions could be included. The addition of 3D models of both Treacher Collins and Goldenhar infants would be beneficial to expand the scope of the program and allow users to further explore and compare the syndromes.

Because of its focus on micrognathia, the algorithm developed for this project covers Pierre Robin sequence, Treacher Collins syndrome, and Goldenhar syndrome. It would be beneficial to implement the workflow used in this project to create separate algorithms for each condition. Additionally, the interactive flowchart workflow could be used to create situations for other time-sensitive medical procedures.

Detail within the flowchart, such as size of endotracheal tube, type of video laryngoscope, amount of anesthetic drug, etc. were left out for simplicity. In the future, this algorithm could be expanded with greater detail to feature more visual content to accompany it such as X-rays, photos, and laryngoscopy videos which would provide additional education opportunities.

In future versions of this project, specific patient situations could be added so that the algorithm would serve as both a training tool and simulator. By providing the users with an outcome to each of their decisions (instead of letting them choose the outcome), the users could practice a more true to life scenario. In these situations, the explanations accompanying each step of the algorithm would be removed and the focus would be placed on the decisions the users make and the resulting outcomes.

In an example scenario, the user may choose to perform an awake nasal intubation on an infant

when they first come to the operating room for mandibular distraction surgery. The physician would be prompted to select which size laryngoscope blade to use, and which kind of analgesics or anesthesia (if any). Once these selections were made, the application may present the baby as being too vigorous for an effective intubation. The user would then be promted to decide what to do next: Try to intubate again, try oral intubation, call for help, give general anesthesia, or perform invasive airway access. Depending on what choice the user makes, the algorithm would continue to give the outcomes as the user progresses along the flowchart (Figure 112).

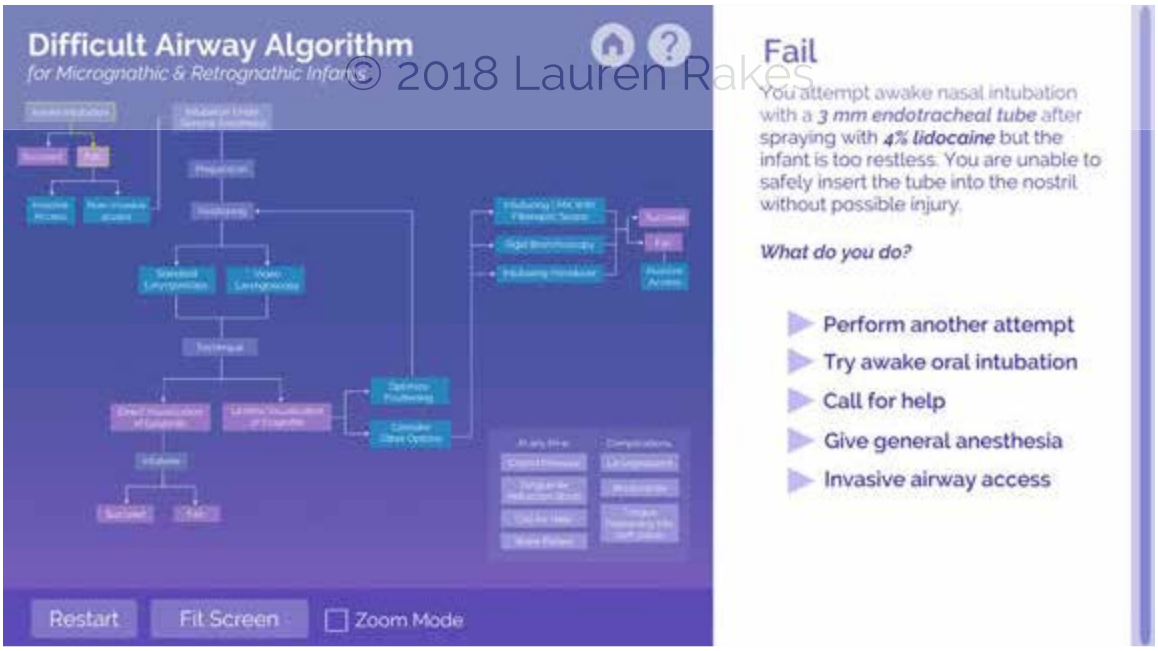


Figure 112. Mockup of future web application mode *(text not intended to be read).*

This application was developed in WebGL format so that it could be widely accessible to a large audience. However, it would be helpful to develop it for use on an iPad (Figure 113) and virtual reality (VR). iPads provide the portability that many computers do not, and they are often used as teaching tools in the classroom and healthcare setting. Virtual reality could be implemented in the 3D Anatomy section of the application, and would allow users to physically navigate around the 3D model and appreciate the scale of the infant's anatomy.

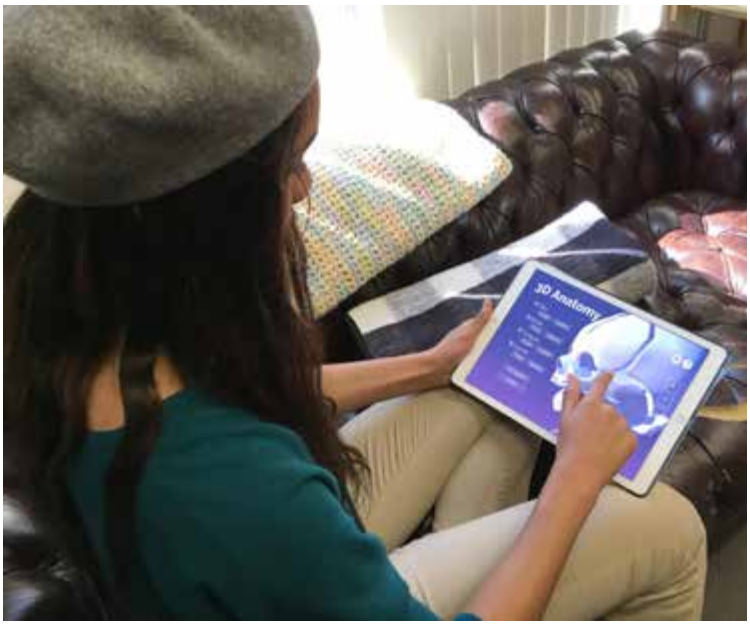


Figure 113. Mockup of iPad app.

Future Integration and Implications for Medical Education

Algorithms are already in use in medical education. However, with the implementation of interactivity, their effectiveness would be improved. By presenting the algorithn within a web application, students and trainees can learn concepts on their own time and review concepts whenever they think it is necessary.

CONCLUSION

Micrognathic infants face risk of injury during airway management due to a lack of experience in managing their unique airway anatomy. This project addresses the gap in training resources for anesthesiologists and otolaryngologists managing micrognathic airways by providing an accessible, interactive, online application made to help improve their knowledge on the options available during the procedure and overall confidence in the operating room. The application provides multiple ways for the physicians to better visualize the anatomy in a 3D space and truly understand the mechanisms behind airway obstruction caused by micrognathia. Additionally, users can learn from a unique interactive algorithm made specifically for this project that explains each step of managing and rescuing a micrognathic infant airway with text, illustrations, and animations.

This web application will be tested through implementation within anesthesiologist education at Johns Hopkins Hospital, and will be modified based on user feedback to improve education effectiveness. The application will provide a foundation for similar projects in other specialities of medicine where quick decisions are critical to the well-being of the patient. Predicted outcomes of using this application include increased familiarity with the anatomy of micrognathic airway of infants, and better decision-making skills when managing the airways of these infants. Users will be able to secure the mental preparation needed to manage the sometimes life-threatening situations that micrognathic pediatric airways can cause. The ultimate goal of this project is to contribute to the decrease in accidental injury to infants during intubation in the operating room.

APPENDICES

Appendix A: Change Scene Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class ChangeScenesALL : MonoBehaviour {

    //This script is attached to a blank “controller object”.
    //The controller can be dragged onto a button.

    public void NextScene (string scene) {
        //When the button with the controller attached is clicked, this script runs.
        //The script looks for a manually defined “string” in the Button inspector.
        //The name of the destination scene is put into the string space.
        //The script loads the scene.
        SceneManager.LoadScene (scene);
    }
}
```

Appendix B: 3D Anatomy Scene Camera Zoom Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Camerazoom : MonoBehaviour {

    // This happens once per frame
    void Update () {
        //If the mouse wheel scrolls up, then this happens
        if (Input.GetAxis (“Mouse ScrollWheel”) > 0){
                //The camera’s field of view gets smaller
                GetComponent<Camera> ().fieldOfView--;
        }

        //If the mouse wheel scrolls down, then this happens
        if (Input.GetAxis (“Mouse ScrollWheel”) < 0){
                //The camera’s field of view gets bigger
                GetComponent<Camera> ().fieldOfView++;
        }
    }
}
```

Appendix C: Algorithm Scene Camera Zoom Script

```
using UnityEngine;
using System.Collections;

public class newzoom : MonoBehaviour
{
    //Zoom speed gets defined manually in inspector
    public float zoomSpeed = 20.0f;

    void Start()
    {
        bool camera_move_enabled = false;
        Debug.Log ("start void");
    }

    void Update ()
    {
        //The camera moves along its Z plane
        float scroll = Input.GetAxis ("Mouse ScrollWheel");
        transform.Translate (0, 0, scroll * zoomSpeed, Space.World);
    }
}
```

Appendix D: Algorithm Scene Camera Pan Script

```
using UnityEngine;
using System.Collections;

public class newpan : MonoBehaviour
{
    //Pan speed gets defined manually in inspector
    public float panSpeed = 1f;

    // Position of cursor when mouse dragging starts
    private Vector3 mouseOrigin;

    // Checks if the camera is being panned
    private bool isPanning;

    void Update ()
    {
        // If right mouse button is held down this happens
        if(Input.GetMouseButtonDown(0))
        {
                //Captures location of where the mouse clicks
                mouseOrigin = Input.mousePosition;
                isPanning = true;
        }

        // If right mouse button is not held down this happens
        if (!Input.GetMouseButton(0)) isPanning=false;

        // If isPanning is true this happens
        if (isPanning){
                // The camera moves on its XY plane
                Vector3 delta = Input.mousePosition - mouseOrigin;
                Vector3 pos = transform.position;
                pos.x += -delta.x * panSpeed;
                pos.y += -delta.y * panSpeed;
                transform.position = pos;
                mouseOrigin = Input.mousePosition;
        }
    }
}
```

Appendix E: Baby Rotate Script

```
using UnityEngine;
using System.Collections;

public class Rotate3 : MonoBehaviour {

    // Speeds are defined privately here
    private float rotationSpeed = 3.0F;
    private float lerpSpeed = 8.0F;
    private Vector3 theSpeed;
    private Vector3 avgSpeed;
    //On start, isDragging is false
    private bool isDragging = false;
    private Vector3 targetSpeedX;

    void Update() {
        //If the right mouse button is clicked, then isDragging becomes true
        if (Input.GetMouseButton (1)) {
                isDragging = true;
        }
        //If the right mouse button is clicked, and isDragging is true, then this happens
        if (Input.GetMouseButton(1) && isDragging) {
                theSpeed = new Vector3(-Input.GetAxis("Mouse X"), Input.GetAxis("Mouse Y"), 0.0F);
                avgSpeed = Vector3.Lerp(avgSpeed, theSpeed, Time.deltaTime * 1);
        }

        else {
                if (isDragging) {
                        theSpeed = avgSpeed;
                        isDragging = false;
                }
                float i = Time.deltaTime * lerpSpeed;
                theSpeed = Vector3.Lerp(theSpeed, Vector3.zero, i);
        }
        //The model rotates
        transform.Rotate(Camera.main.transform.up * theSpeed.x * rotationSpeed, Space.World);
        transform.Rotate(Camera.main.transform.right * theSpeed.y * rotationSpeed, Space.World);
    }
}
```

Appendix F: Baby Pan Script

```
using UnityEngine;
using System.Collections;

public class panbaby3 : MonoBehaviour {

    //Drag speed gets defined manually in inspector
    public float dragSpeed = 1f;
    Vector3 lastMousePos;

    //Captures location of where the mouse clicks
     void OnMouseDown() {
          lastMousePos = Input.mousePosition;
          }

    //Calculates drag based on mouse down position, current mouse position, and drag speed
     void OnMouseDrag() {
          Vector3 delta = Input.mousePosition - lastMousePos;
          Vector3 pos = transform.position;
          pos.x += delta.x * dragSpeed;
          pos.y += delta.y * dragSpeed;
          transform.position = pos;
          lastMousePos = Input.mousePosition;
          }
}
```

Appendix G: Change Materials (Skin) Script

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class ChangeMaterialSkin : MonoBehaviour {

    // The first material applied to the object is defined here
    public Material Mat1;
    // The second material applied to the object is defined here
    public Material Mat2;
    // A bool condition is set where the first material is applied to the object upon application start, and the second material is not
    public bool FirstMaterial = true;
    public bool SecondMaterial = false;

    // The public variables are defined, so they can be edited in the Inspector
    // This is the button that will initiate the material change
    public Button yourbutton;
    // This is the full skin object
    public GameObject Wholeskin;
    // This is the sagittal cross-section skin object
    public GameObject Halfskin;

    // Initialization
    void Start () {
        // When this button is clicked, TaskOnClick happens
        yourbutton.onClick.AddListener (TaskOnClick);
        // The skin objects will have material 1 applied upon start
        Wholeskin.GetComponent<Renderer>().material = Mat1;
        Halfskin.GetComponent<Renderer>().material = Mat1;
    }

    // Anything under TaskOnClick happens when the button is clicked
    void TaskOnClick (){
        // If the first material is already applied to the skin objects, this happen
        if (FirstMaterial) {
                // The material applied to both skin objects changes to material 2
                Wholeskin.GetComponent<Renderer>().material = Mat2;
                Halfskin.GetComponent<Renderer>().material = Mat2;
                // This adjusts the renderer so it does not cast shadows on the bones
```

```
                Wholeskin.GetComponent<MeshRenderer>().shadowCastingMode = UnityEngine.
Rendering.ShadowCastingMode.Off;
                Halfskin.GetComponent<MeshRenderer>().shadowCastingMode = UnityEngine.
Rendering.ShadowCastingMode.Off;
                // The bool changes, material 2 is now true, material 1 is false
                SecondMaterial = true;
                FirstMaterial = false;

    // If the second material is already applied to the skin objects, then this happens
    } else if (SecondMaterial) {
                // The material applied to both skin objects changes to material 1
                Wholeskin.GetComponent<Renderer>().material = Mat1;
                Halfskin.GetComponent<Renderer>().material = Mat1;
                // This adjusts the renderer of material 1 so it can cast shadows upon itself
                Wholeskin.GetComponent<MeshRenderer>().shadowCastingMode = UnityEngine.
Rendering.ShadowCastingMode.On;
                Halfskin.GetComponent<MeshRenderer>().shadowCastingMode = UnityEngine.
Rendering.ShadowCastingMode.On;
                // The bool changes, material 1 is now true, material 2 is false
                FirstMaterial = true;
                SecondMaterial = false;
        }

        // If the button has been clicked, the following will print out in the Console
        Debug.Log ("You have clicked the button");
    }
}
```

Appendix H: Change to Sagittal (Skin) Script

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class SkinSag : MonoBehaviour {

    //The public variables are defined, so they can be edited in the Inspector
    public bool Fullskin = true;
    public bool Halfskin = false;
    public GameObject fullskin;
    public GameObject halfskin;
    public Button yourbuttonskinsag;
    public Toggle toggle;

    void Start () {
        Button SkinBtnSag = yourbuttonskinsag.GetComponent<Button> ();
        // When this button is clicked, TaskOnClick2 happens
        SkinBtnSag.onClick.AddListener (TaskOnClick2);
        fullskin.GetComponent<MeshRenderer> ().enabled = true;
        halfskin.GetComponent<MeshRenderer> ().enabled = true;
    }

    void TaskOnClick2 (){
        toggle.isOn = true;
        // If the full skin model is on, this happens
        if (Fullskin) {
                fullskin.SetActive (true);
                halfskin.SetActive (true);
                //The renderer of the full skin model turns off
                //The renderer of the sagittal skin model turns on
                fullskin.GetComponent<MeshRenderer> ().enabled = false;
                halfskin.GetComponent<MeshRenderer> ().enabled = true;
                Halfskin = true;
                Fullskin = false;

        // If the sagittal skin model is on, this happens
        } else if (Halfskin) {

                halfskin.SetActive (true);
                fullskin.SetActive (true);
```

```
                    //The renderer of the full skin model turns on
                    //The renderer of the sagittal skin model turns off
                    fullskin.GetComponent<MeshRenderer> ().enabled = true;
                    halfskin.GetComponent<MeshRenderer> ().enabled = false;
                    Fullskin = true;
                    Halfskin = false;
            }
        }
    }
```

Appendix I: Baby Model Rotation Buttons Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class RotateButtonsAnt : MonoBehaviour {

    //The public variables are defined, so they can be edited in the Inspector
    public Button Anterior;
    public Button Posterior;
    public Button Left;
    public Button Right;

    void Start () {
        // When these buttons are clicked, these TaskOnClick events happen
        Button antbtn = Anterior.GetComponent<Button> ();
        antbtn.onClick.AddListener (TaskOnClick);
        Button posbtn = Posterior.GetComponent<Button> ();
        posbtn.onClick.AddListener (TaskOnClick2);
        Button leftbtn = Left.GetComponent<Button> ();
        leftbtn.onClick.AddListener (TaskOnClick3);
        Button rightbtn = Right.GetComponent<Button> ();
        rightbtn.onClick.AddListener (TaskOnClick4);
    }

    // The baby model snaps to these rotation values when the buttons are clicked
    void TaskOnClick () {
        transform.eulerAngles = new Vector3 (-178.3f, -1.61f, 0f);
    }
    void TaskOnClick2 () {
        transform.eulerAngles = new Vector3 (-186.8f, 177.21f, 0.1f);
    }
    void TaskOnClick3 () {
        transform.eulerAngles = new Vector3 (-183.2f, 84.9f, -4.2f);
    }
    void TaskOnClick4 () {
        transform.eulerAngles = new Vector3 (-182.18f, -84.54f, 4.3f);
    }
}
```

Appendix J: Fit Screen Script

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class CameraMoveAlgorithm3 : MonoBehaviour {

    //The public variables are defined, so they can be edited in the Inspector
    public GameObject TargetPositionreset;
    public int speed = 2;
    public bool uh0 = false;
    public Button fitscreen;

    void Start() {
        // When this button is clicked, TaskOnClick happens
        fitscreen.onClick.AddListener (TaskOnClick0);
    }

    // If any of these buttons are clicked, then the camera action stops
    void Update() {
        if(Input.GetMouseButtonDown(1)) {
                uh0 = false;
        }
        if(Input.GetMouseButtonDown(0)) {
                uh0 = false;
        }
        else if(Input.GetKey(KeyCode.RightArrow)) {
                uh0 = false;
        }
        else if(Input.GetKey(KeyCode.LeftArrow)) {
                uh0 = false;
        }
        else if(Input.GetKey(KeyCode.UpArrow)) {
                uh0 = false;
        }
        else if(Input.GetKey(KeyCode.DownArrow)) {
                uh0 = false;
        }
        else if (Input.GetAxis("Mouse ScrollWheel") > 0) {
                uh0 = false;
        }
```

```
        else if (Input.GetAxis("Mouse ScrollWheel") < 0){
            uh0 = false;
        }
        // If this is true, the camera moves
        else if (uh0) {
            Camera.main.transform.position = Vector3.Lerp(transform.position, TargetPositionreset.
transform.position, speed * Time.deltaTime);
            Camera.main.transform.rotation = Quaternion.Lerp(transform.rotation,
TargetPositionreset.transform.rotation, speed * Time.deltaTime);
        }
    }

    void TaskOnClick0 () {
        Camera.main.fieldOfView = 32.4f;
        uh0 = true;
    }
}
```

Appendix K: Invoke Button Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class InvokeScript : MonoBehaviour {

    //The public variables are defined, so they can be edited in the Inspector
    //This is the button that will link to the other button and perform its action
    public Button myButton;
    //This is the target button that already performs the desired action
    public Button targetbutton;


    // Initialization
    void Start () {
        //When myButton is clicked, TaskOnClick happens
        myButton.onClick.AddListener (TaskOnClick);
    }

    // Update is called once per frame
    void TaskOnClick () {
        //targetbutton is clicked
        targetbutton.onClick.Invoke();
    }
}
```

Appendix L: Disable Scripts Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Disablescriptbehindpanel : MonoBehaviour {

    //The public variables are defined, so they can be edited in the Inspector
    public newpan newpanscript;
    public newzoom newzoomscript;

    // On start, the scripts are true
    void Start () {
        newpanscript =  Camera.main.GetComponent<newpan>();
        newzoomscript =  Camera.main.GetComponent<newzoom>();
        newpanscript.enabled = true;
        newzoomscript.enabled = true;
    }

    // When the mouse enters the Box Collider, the scripts are false
    void OnMouseOver()  {
        newpanscript =  Camera.main.GetComponent<newpan>();
        newzoomscript =  Camera.main.GetComponent<newzoom>();
        newpanscript.enabled = false;
        newzoomscript.enabled = false;
    }

    //When the mouse exits the Box Collider, the scripts are true again
    void OnMouseExit() {
        newpanscript =  Camera.main.GetComponent<newpan>();
        newzoomscript =  Camera.main.GetComponent<newzoom>();
        newpanscript.enabled = true;
        newzoomscript.enabled = true;
    }
}
```

References

1. A., Judith, Jeroen Hermanides, Peter L., Jasper J., and David R. “Bradycardia in Children During General Anaesthesia.” Cardiac Arrhythmias - New Considerations, 2012. doi:10.5772/35752.

2. ”Airway Management (Anesthesia Text).” Open Anesthesia. 2018. Accessed January 2018. http://www.openanesthesia.org/airway_management_anesthesia_text/.

3. “Airway Management Training Manikin / Infant / Head / Portable AirSim® Pierre Robin X TruCorp.” MedicalExpo - The Online Medical Device Exhibition. 2018. Accessed March 2018. http://www.medicalexpo.com/prod/trucorp/product-89869-683945.html.

4. Airway Management Education Center. “The Difficult Airway App.” Apple App Store, Vers. 2.2 (2018). https://itunes.apple.com/us/app/the-difficult-airway-app/id433176317?mt=8

5. “Airway: Pediatric vs Adult.” OpenAnesthesia. 2018. Accessed February 2018. http://www.openanesthesia.org/airway_pediatric_vs_adult/.

6. Apfelbaum, Jeffrey L., Carin A. Hagberg, Robert A. Caplan, Casey D. Blitt, Richard T. Connis, David G. Nickinovich, Carin A. Hagberg, Robert A. Caplan, Jonathan L. Benumof, Frederic A. Berry, Casey D. Blitt, Robert H. Bode, Frederick W. Cheney, Richard T. Connis, Orin F. Guidry, David G. Nickinovich, and Andranik Ovassapian. “Practice Guidelines for Management of the Difficult Airway.” Anesthesiology 118, no. 2 (2013): 251-70. doi:10.1097/aln.0b013e31827773b2.

7. Cladis, Franklyn, Anand Kumar, Lorelei Grunwaldt, Todd Otteson, Matthew Ford, and Joseph E. Losee. “Pierre Robin Sequence.” Anesthesia & Analgesia 119, no. 2 (March 2014): 400-12. doi:10.1213/ane.0000000000000301.

8. Clark, H.brent. “Pediatric neuropathology.” Pediatric Neurology 13, no. 1 (1995): 86. doi:10.1016/0887-8994(95)90022-5.

9. Culatta, Richard. “Andragogy (Malcolm Knowles).” Andragogy. 2015. Accessed January 2018. http://www.instructionaldesign.org/theories/andragogy.html.

10. Culatta, Richard. “Experiential Learning (Carl Rogers).” Experiential Learning. 2015. Accessed January 2018. http://www.instructionaldesign.org/theories/experiential-learning.html.

11. Culatta, Richard. “Minimalism (J.M. Carroll).” Minimalism. 2015. Accessed January 2018. http://www.instructionaldesign.org/theories/minimalism.html.

12. Drummond, Gordon B. “Spontaneous breathing during anaesthesia: first, do no harm.” Signa Vitae - A Journal In Intensive Care And Emergency Medicine 2, no. 2 (2007): 6-9. doi:10.22514/sv22.102007.1.

13. Flores, Roberto. “Neonatal Mandibular Distraction Osteogenesis.” Seminars in Plastic Surgery 28, no. 04 (November 2014): 199-206. doi:10.1055/s-0034-1390173.

14. “Goldenhar disease.” Genetic and Rare Diseases Information Center. May 2017. Accessed February 2018. https://rarediseases.info.nih.gov/diseases/6540/goldenhar-disease.

15. Hampson-Evans, Darryl, Patrick Morgan, and Mark Farrar. “Pediatric laryngospasm.” Pediatric Anesthesia 18, no. 4 (February 29, 2008): 303-07. doi:10.1111/j.1460-9592.2008.02446.x.

16. Heggie, Andrewa, Ricky Kumar, and Jocelynm Shand. “The role of distraction osteogenesis in the management of craniofacial syndromes.” Annals of Maxillofacial Surgery 3, no. 1 (April 2013): 4-10. doi:10.4103/2231-0746.110063.

17. Huang, Faye , Lun-Jou Lo, Yu-Ray Chen, Johnson Yang, Chen-Kuang Niu, and Mei-Yung Chung. “Tongue-Lip Adhesion in the Management of Pierre Robin Sequence with Airway Obstruction: Technique and Outcome.” Chang Gung Med J 28 (February 2005): 90-96. Accessed February 2018. https://www.ncbi.nlm.nih.gov/pubmed/15880984.

18. “Isolated Pierre Robin sequence - Genetics Home Reference.” U.S. National Library of Medicine. March 6, 2018. Accessed February 2018. https://ghr.nlm.nih.gov/condition/isolated-pierre-robin-sequence.

19. Jablonski, Anita M., Anna R. Dupen, and Mary Ersek. “The Use of Algorithms in Assessing and Managing Persistent Pain in Older Adults.” AJN, American Journal of Nursing 111, no. 3 (February 19, 2011): 34-43. doi:10.1097/10.1097/01.naj.0000395239.60981.2f.

20. Jagannathan, Narasimhan, and David T. Wong. “Successful Tracheal Intubation through an Intubating Laryngeal Airway in Pediatric Patients with Airway Hemorrhage.” The Journal of Emergency Medicine 41, no. 4 (2011): 369-73. doi:10.1016/j.jemermed.2010.05.066.

21. Jennifer Whitlock, RN, MSN, FN | Reviewed by Richard N. Fogoros, MD. “Why Intubation Is Required for Most Surgeries.” Verywell Health. December 7, 2017. Accessed February 2018. https://www.verywell.com/what-is-intubation-and-why-is-it-done-3157102.

22. Komaroff, A. L. “Algorithms and the “art of medicine.” American Journal of Public Health 72, no. 1 (1982): 10-12. doi:10.2105/ajph.72.1.10.

23. Level Ex, Inc. “Airway Ex.” Apple App Store, Vers. 1.4.9 (2018). https://itunes.apple.com/us/app/airway-ex/id1154656060?mt=8

24. Lingnau, A., M. Kuhn, A. Harrer, D. Hofmann, M. Fendrich, and H.u. Hoppe. “Enriching traditional classroom scenarios by seamless integration of interactive media.” Proceedings 3rd IEEE International Conference on Advanced Technologies, July 2003, 135-39. doi:10.1109/icalt.2003.1215043.

25. Malherbe, Stephan, Simon Whyte, Permendra Singh, Erica Amari, Ashlee King, and J. Mark Ansermino. “Total intravenous anesthesia and spontaneous respiration for airway endoscopy in children - a prospective evaluation.” Pediatric Anesthesia 20, no. 5 (2010): 434-38. doi:10.1111/j.1460-9592.2010.03290.x.

26. Nguyen, Linh T., Sudip D. Thakar, Angela T. Truong, and Dam-Thuy Truong. "Orotracheal Intubation Using the Retromolar Space: A Reliable Alternative Intubation Approach to Prevent Dental Injury." Case Reports in Anesthesiology 2016 (November 2016): 1-3. doi:10.1155/2016/3529415.

27. Nicastri, Daniel G., and Todd S. Weiser. "Rigid Bronchoscopy: Indications and Techniques." Operative Techniques in Thoracic and Cardiovascular Surgery 17, no. 1 (2012): 44-51. doi:10.1053/j.optechstcvs.2012.03.001.

28. Nickson, Chris. "Paediatric Airway." LITFL • Life in the Fast Lane Medical Blog. September 21, 2017. Accessed January 2018. https://lifeinthefastlane.com/ccc/paediatric-airway/.

29. "Pediatric Airway." Department of Pediatrics. 2018. Accessed January 2018. http://www.pediatrics.wisc.edu/education/sedation-program/sedation-education/pediatric-airway.

30. Qaqish, Clement, and John F. Caccamese. "The tongue-lip adhesion." Operative Techniques in Otolaryngology-Head and Neck Surgery 20, no. 4 (2009): 274-77. doi:10.1016/j.otot.2009.10.020.

31. Rasch, Deborah K., Frederick Browder, Marjory Barr, and Donald Greer. "Anaesthesia for Treacher Collins and Pierre Robin syndromes: A report of three cases." Canadian Anaesthetists' Society Journal 33, no. 3 (1986): 364-70. doi:10.1007/bf03010751.

32. Rawlinson, Ellen. "Postoperative Airway Complications after Cleft Palate Repair." Anaesthesia Tutorial of the Week. August 22, 2011. Accessed January 2018. http://www.bing.com/cr?IG=EF89F3433E9447D0A2D4172F97218D98&CID=1233DEEFF5B86E1716B4D542F4176F2F&rd=1&h=uYimxU1o3-KUzO983eKk8itJvLoUSoLs53-Dr39ks9o&v=1&r=http%3a%2f%2fe-safe-anaesthesia.org%2fe_library%2f14%2fPostoperative_Airway_Obstruction_after_Cleft_Palate_Surgery_TOTW_237_2011.pdf&p=DevEx,5066.1.

33. Reichman, Eric F. "Endotracheal Tube Intubating Introducers and Bougies." In Emergency Medicine Procedures. 2nd ed. New York, NY: McGraw-Hill, 2013.

34. Romig, Lou. "JumpSTART and MCI Triage." The JumpSTART Pediatric MCI Triage Tool. May 29, 2012. Accessed January 2018. http://www.jumpstarttriage.com/JumpSTART_and_MCI_Triage.php.

35. Roy, Soham, Irving Basañez, and Rhonda Alexander. "Performance of Rigid Bronchoscopy." Anesthesia Key. July 06, 2016. Accessed February 2018. https://aneskey.com/bronchoscopy-2/.

36. Salvatore, Rosanne. "Would You Survive The Oregon Trail?" Bustle. February 22, 2018. Accessed February 2018. http://www.bustle.com/flowcharts/would-you-survive-the-oregon-trail-44081.

37. Savkovic, Admedina, Jasmin Delic, Eldar Isakovic, Farid Ljuca, and Sabina Nuhbegovic. "Age Characteristics of the Larynx in Infants During the First Year of Life." Periodicum Biologorum 112, no. 1 (2010): 75-82. https://hrcak.srce.hr/file/80577.

38. Saxena, Kirti. N., H. Nischal, M. Bhardwaj, and B. Shastry. "Right molar intubation in a child with Pierre Robin syndrome, cleft palate and tongue tie." BJA: British Journal of Anaesthesia 99, no. ELetters Supplement (February 2008): 141-42. doi:10.1093/bja/el_2199.

39. Schwarz, Daniel, Petr Štourač, Martin Komenda, Hana Harazim, Martina Kosinová, Jakub Gregor, Richard Hůlek, Olga Smékalová, Ivo Křikava, Roman Štoudek, and Ladislav Dušek. "Interactive Algorithms for Teaching and Learning Acute Medicine in the Network of Medical Faculties MEFANET." Journal of Medical Internet Research 15, no. 7 (February 08, 2013). doi:10.2196/jmir.2590.

40. Sloas, Andrew. "Pediatric Airway 101." Pediatric Emergency Medicine and Educational and Directional Podcast (audio blog), March 2012. Accessed January 2018. http://www.pemed.org/blog/2012/3/3/pediatric-airway-101.html.

41. Szulewski, Adam, and Bob McGraw. "Step 2: Position Head and Neck." Advanced Airway Management. Accessed February 2018. https://meds.queensu.ca/central/assets/modules/advanced-airway-management/step_2_position_head_and_neck.html.

42. "Treacher Collins Syndrome." Craniofacial Disorders. February 2016. Accessed January 2018. http://www.faces-cranio.org/Disord/Treacher.htm.

43. "Treacher Collins syndrome - Genetics Home Reference." U.S. National Library of Medicine. March 6, 2018. Accessed February 2018. https://ghr.nlm.nih.gov/condition/treacher-collins-syndrome.\

44. "Unity User Manual (2017.3)." Unity - Manual: Unity User Manual (2017.3). 2018. Accessed November 2017. https://docs.unity3d.com/Manual/index.html.

Vita

Lauren Rakes was born in Charlottesville, VA and grew up surrounded by creative and artistic family members. She had always been fascinated by science and nature, but chose to attend Virginia Commonwealth University in Richmond, Virginia to get a degree in Art Education. Instead, she discovered the Scientific and Preparatory Medical Illustration program and fell in love. Her new major combined her interests and allowed her to educate through visual storytelling. She graduated summa cum laude in May 2016 with a BFA in Communication Arts, a concentration in "SciMed' Illustration, and a minor in Biology.

Lauren began her graduate studies in the Medical and Biological Illustration program in the Department of Art as Applied to Medicine at the Johns Hopkins University School of Medicine in 2016. Her goal was to learn more about science and medicine and how to communicate those subjects to the world around her. While studying at Hopkins, she has had the pleasure of learning under accomplished medical illustrators and working as a team with physicians, scientists, and her fellow classmates. She received the Frank H. Netter Memorial Scholarship for her academic performance at Johns Hopkins University in 2017. Additionally, her artwork was recognized at the 2017 annual Association of Medical Illustrators meeting, where she earned an Award of Excellence in the student anatomical/pathological category. She was honored to have been awarded the Vesalius Trust's top research scholarship, the 2018 Alan Cole Award, on the proposal for this thesis project. Lauren is currently a candidate to receive a Master of Arts in Medical and Biological Illustration in May, 2018.